

[illegible][illegible]

(1)	136	DECLARATIONS
(1)	287	REGISTER DEFINITIONS
(1)	346	CONTROLLER INITIALIZATION
(1)	427	UNIT INITIALIZATION
(1)	512	MAINTENANCE ROUTINES
(1)	577	OUTPUT MODEM CONTROL
(1)	631	DZ-11 MODEM POLLER
(1)	717	RECEIVER INTERRUPT SERVICE
(1)	891	START I/O ROUTINE
(1)	943	PORT ROUTINES STOP, RESUME, XON, XOFF
(1)	1085	OUTPUT INTERRUPT SERVICE
(1)	1348	SET SPEED, PARITY PARAMETERS
(1)	1400	INITIALIZE DZ-11 MODEM POLLING

```
0000 1 .if df DZV
0000 2 .TITLE DZVDRIVER - Port Driver for DZV-11 support
0000 3 .iff
0000 4 .TITLE DZDRIVER - Port Driver for DZ-11 support
0000 5 .endc
0000 6 .IDENT 'V04-000'
0000 7
0000 8
0000 9 *****
0000 10 *
0000 11 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 12 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 13 * ALL RIGHTS RESERVED.
0000 14 *
0000 15 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 16 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 17 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 18 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 19 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 20 * TRANSFERRED.
0000 21 *
0000 22 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 23 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 24 * CORPORATION.
0000 25 *
0000 26 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 27 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 28 *
0000 29 *
0000 30 *****
0000 31
0000 32 ++
0000 33 FACILITY:
0000 34
0000 35 VAX/VMS TERMINAL DRIVER
0000 36
0000 37 ABSTRACT:
0000 38
0000 39 DZ PORT DRIVER
0000 40 This module functions as a port driver for DZ11 and DZ32 terminal
0000 41 controllers. It contains hardware specific port level service
0000 42 routines.
0000 43
0000 44 AUTHOR:
0000 45
0000 46 RICK SPITZ
0000 47
0000 48 Revision history:
0000 49
0000 50 V03-028 M1R0480 Michael I. Rosenblum 8-Aug-1984
0000 51 Fix bugs found in testing the DZ-32 and reported in
0000 52 QAR 1126 (FT1).
0000 53
0000 54 V03-027 LMP0275 L. Mark Pilant, 12-Jul-1984 21:01
0000 55 Initialize the ACL info in the ORB to be a null descriptor
0000 56 list rather than an empty queue. This avoids the overhead
0000 57 of locking and unlocking the ACL mutex, only to find out
```

```
0000 58 : that the ACL was empty.
0000 59 :
0000 60 : V03-026 EMD0090 Ellen M. Dusseault 30-Apr-1984
0000 61 : Add DEV$M_NNM characteristic to DEVCHAR2 so that these
0000 62 : devices will have the 'node$' prefix.
0000 63 :
0000 64 : V03-025 LMP0221 L. Mark Pilant, 7-Apr-1984 13:37
0000 65 : Change UCBSL_OWNUIC to ORBSL_OWNER and UCBSW_VPROT to
0000 66 : ORBSW_PROT.
0000 67 :
0000 68 : V03-024 JLV0321 Jake VanNoy 5-JAN-1984
0000 69 : Minor enhancements to DZV11 support. Comment out
0000 70 : DPT_STORE for parity, must fix a restriction.
0000 71 :
0000 72 : V03-023 MIR0055 Michael I. Rosenblum 30-June-1983
0000 73 : Remove code from unit and controller-init routines
0000 74 : and make insert calls to class driver macros
0000 75 : Add DZV11 support.
0000 76 :
0000 77 : V03-022 RKS0022 RICK SPITZ 14-MAR-1983
0000 78 : ADD ENHANCEMENTS TO SUPPORT LOGICAL UCB.
0000 79 :
0000 80 : V03-021 MIR0022 Michael I. Rosenblum 19-Jan-1982
0000 81 : Change references to UCBSB_ERTCNT to use UCBSW_IT_UNITBIT
0000 82 : to be more maintainable.
0000 83 : Replace old vector table with new vector table macro.
0000 84 : Remove references to UCBSL_DEVDEPEND and UCBSQ_IT_STATE
0000 85 : move these references into the class driver jacket routines
0000 86 :
0000 87 : V03-020 MIR0021 Michael I. Rosenblum 17-Jan-1983
0000 88 : Fix DZ32 DZ_SET table entry.
0000 89 :
0000 90 : V03-019 RKS0019 RICK SPITZ 13-JAN-1983
0000 91 : Repair problem with port vector macro
0000 92 :
0000 93 : V03-018 MIR0019 Michael I. Rosenblum 11-Jan-1982
0000 94 : Fix undefined symbol created by MIR0018.
0000 95 :
0000 96 : V03-017 MIR0018 Michael I. Rosenblum 07-Jan-1983
0000 97 : Change the port vector table to use the $VEC macros
0000 98 :
0000 99 : V03-016 MIR0017 Michael I. Rosenblum 05-Jan-1983
0000 100 : Add powerfail check in the Unit init routine to allow the
0000 101 : terminal class driver to take positive action on powerfail.
0000 102 : Change code to accept a byte value as returns from the
0000 103 : GETNXT and PUTNXT class services, this will remove this
0000 104 : information from the condition codes.
0000 105 :
0000 106 : V03-015 MIR0016 Michael I. Rosenblum 29-Dec-1982
0000 107 : Replace time calculation code with TIMSET macro call
0000 108 :
0000 109 : V03-014 MIR0015 Michael I. Rosenblum 20-Dec-1982
0000 110 : Remove entry point to reflect the redefinition of the PORT_DISCONNECT
0000 111 : entry point.
0000 112 :
0000 113 : V03-013 MIR0014 Michael I. Rosenblum 17-Dec-1982
0000 114 : Remove code to calculate flow control characters from
```

```
0000 115 : port XON and XOFF routines and move that code into
0000 116 : the class driver.
0000 117 :
0000 118 : V03-012 RKS0012 RICK SPITZ 16-SEP-1982
0000 119 : Check reference count in unit init to determine if
0000 120 : modem control should be initialized or hungup. This
0000 121 : is needed to insure that a hangup ^Y is posted on
0000 122 : powerfail for modem lines.
0000 123 :
0000 124 : V03-011 JLV0211 Jake VanNoy 2-JUL-1982
0000 125 : Remove check for powerfail in unit init that prevents
0000 126 : SETUP_UCB from being called. This insures that UCB fields
0000 127 : are initialized correctly when Unit init is called for
0000 128 : use with CSS unibus switch.
0000 129 :
0000 130 : V03-010 KDM0002 Kathleen D. Morse 28-Jun-1982
0000 131 : Added $DEVDEF, $IPLDEF, $PRDEF, and $SSDEF.
0000 132 :
0000 133 :
0000 134 :--
```

```
0000 136          .SBTTL  DECLARATIONS
0000 137
0000 138          :
0000 139          : EXTERNAL DEFINITIONS:
0000 140          :
0000 141          $ACBDEF          : DEFINE ACB
0000 142          $CRBDEF          : DEFINE CRB
0000 143          $DCDEF           : DEVICE DEFINITIONS
0000 144          $DDBDEF          : DEFINE DDB
0000 145          $DEVDEF          : DEFINE DEVICE TYPES
0000 146          $DYNDEF          : DYNAMIC STRUCTURE DEFINITIONS
0000 147          $IDBDEF          : DEFINE IDB OFFSETS
0000 148          $IODEF           : DEFINE I/O FUNCTION CODES
0000 149          $IPLDEF          : DEFINE INTERRUPT PRIORITY LEVELS
0000 150          $IRPDEF          : IRP DEFINITIONS
0000 151          $ORBDEF          : DEFINE OBJECT'S RIGHTS BLOCK OFFSETS
0000 152          $PRDEF           : DEFINE PROCESSOR REGISTERS
0000 153          $SSDEF           : DEFINE SYSTEM STATUS CODES
0000 154          $TTYDEF          : DEFINE TERMINAL DRIVER SYMBOLS
0000 155          $TTDEF           : DEFINE TERMINAL TYPES
0000 156          $TT2DEF          : DEFINE EXTENDED DEFINITIONS
0000 157          $TQDEF           : DEFINE TIMER QUEUE OFFSETS
0000 158          $UCBDEF          : DEFINE UCB
0000 159          $UBADEF          : DEFINE UBA
0000 160          $VECDEF          : DEFINE VECTOR FOR CRB
0000 161          $TTYMACS         : DEFINE TERMINAL DRIVER MACROS
0000 162          $TTYDEFS         : DEFINE TERMINAL DRIVER SYMBOLS
0000 163          $TTYMODEM        : DEFINE MODEM DEFINITIONS
0000 164
0000 165
0000 166          :
0000 167          : LOCAL STORAGE
0000 168          :
0000 169          .PSECT $$$105_PROLOGUE
0000 170
0000 171          :
0000 172          : Driver prologue table:
0000 173          :
0000 174
0000 175 DZSDPT::          : DRIVER START
0000 176 .IF DF DZV
0000 177     DPTAB          : DRIVER PROLOGUE TABLE
0000 178     -              : End and offset to INIT's vectors
0000 179     END=DZ$END,-    : SIZE OF UCB
0000 180     UCBSIZE=UCB$C TT LENGTH,-
0000 181     FLAGS=DPT$M_NOUNLOAD,- : DO NOT ALLOW UNLOAD
0000 182     ADAPTER=UBA,-    : ADAPTER TYPE
0000 183     DEFUNITS=4,-     : DZV has 4 units
0000 184     NAME=DZDRIVER,-  : NAME OF DRIVER
0000 185     VECTOR=PORT_VECTOR : PORT DRIVER VECTOR TABLE
0000 186 .IFF
0000 187     DPTAB          : DRIVER PROLOGUE TABLE
0000 188     -              : End and offset to INIT's vectors
0000 189     END=DZ$END,-    : SIZE OF UCB
0000 190     UCBSIZE=UCB$C TT LENGTH,-
0000 191     FLAGS=DPT$M_NOUNLOAD,- : DO NOT ALLOW UNLOAD
0000 192     ADAPTER=UBA,-    : ADAPTER TYPE
0000 193     DEFUNITS=8,-     : Number of units to create
0000 194     NAME=DZDRIVER,-  : NAME OF DRIVER
```

```
0000 193 VECTOR=PORT_VECTOR ; PORT DRIVER VECTOR TABLE
0038 194 .ENDC
0038 195 DPT_STORE INIT
0038 196 DPT_STORE UCB,UCB$B_FIPL,B,8 ; FORK IPL
003C 197 DPT_STORE UCB,UCB$B_DEVCHAR,L,<- ; CHARACTERISTICS
003C 198 DEV$M_REC:-
003C 199 DEV$M_AVL:-
003C 200 DEV$M_IDV:-
003C 201 DEV$M_ODV:-
003C 202 DEV$M_TRM:-
003C 203 DEV$M_CCL>
0043 204 DPT_STORE UCB,UCB$B_DEVCHAR2,L,- ; Device Characteristics
0043 205 <DEV$M_NNM> ; prefix with 'node$'
004A 206 DPT_STORE UCB,UCB$B_DEVCLASS,B,DC$ TERM;
004E 207 DPT_STORE UCB,UCB$B_TT_DETYPE,B,TT$ UNKNOWN ; TYPE
0052 208 DPT_STORE UCB,UCB$B_TT_DESIZE,B,TT$GW_DEFBUF ; BUFFER SIZE
0059 209 DPT_STORE UCB,UCB$B_TT_DECHAR,B,TT$GL_DEFCHAR ; DEFAULT CHARACTERS
0060 210 DPT_STORE UCB,UCB$B_TT_DECHA1,B,TT$GL_DEFCHAR2 ; DEFAULT CHARACTERS
0067 211 DPT_STORE UCB,UCB$B_TT_DESPEE,B,TT$GB_DEFSPEED ; DEFAULT SPEED
006E 212 DPT_STORE UCB,UCB$B_TT_DESPEE+1,B,TT$GB_RSPEED ; DEFAULT SPEED
0075 213 DPT_STORE UCB,UCB$B_TT_DEPARI,B,TT$GB_PARITY ; DEFAULT PARITY
007C 214 ; DPT_STORE UCB,UCB$B_TT_PARITY,B,TT$GB_PARITY ; DEFAULT PARITY
007C 215 DPT_STORE UCB,UCB$B_DEVTYPE,B,TT$ UNKNOWN ; TYPE
0080 216 DPT_STORE UCB,UCB$B_DEVBUFSIZ,B,TT$GW_DEFBUF ; BUFFER SIZE
0087 217 DPT_STORE UCB,UCB$B_DEVDEPEND,B,TT$GL_DEFCHAR ; DEFAULT CHARACTERS
008E 218 DPT_STORE UCB,UCB$B_DEVDEPND2,B,TT$GL_DEFCHAR2 ; DEFAULT CHARACTERS
0095 219 DPT_STORE UCB,UCB$B_TT_SPEED,B,TT$GB_DEFSPEED ; DEFAULT SPEED
009C 220 DPT_STORE UCB,UCB$B_TT_SPEED+1,B,TT$GB_RSPEED ; DEFAULT SPEED
00A3 221 DPT_STORE UCB,UCB$B_DIPL,B,21 ; DEVICE IPL
00A7 222 DPT_STORE UCB,UCB$B_TT_WFLINK,L,0 ; Zero write queue.
00AE 223 DPT_STORE UCB,UCB$B_TT_WBLINK,L,0 ; Zero write queue.
00B5 224 DPT_STORE UCB,UCB$B_TT_RTIMOU,L,0 ; Zero read timed out disp.
00BC 225 DPT_STORE ORB,ORB$B_FLAGS,B,- ; Protection block flags
00BC 226 <ORB$M_PROT 16> ; SOGW protection word
00C0 227 DPT_STORE ORB,ORB$B_PROT,B,TT$GW_PROT ; Default allocation protection
00C7 228 DPT_STORE ORB,ORB$B_OWNER,B,TT$GL_OWNUIC ; Default owner UIC
00CE 229 DPT_STORE DDB,DDB$B_DDT,D,DZ$DDT
00D3 230
00D3 231 DPT_STORE REINIT
00D3 232 DPT_STORE CRB,CRB$B_INTD+VECSL_INITIAL,D,DZ$INITIAL ; CONTROLLER INIT
00D8 233 DPT_STORE CRB,CRB$B_INTD+VECSL_UNITINIT,D,DZ$INITLINE ; UNIT INIT
00DD 234 DPT_STORE END
0000 235
0000 236 DDTAB DEVNAM = DZ,- ; DUMMY DZ PORT DRIVER DISPATCH TABLE
0000 237 START = 0,-
0000 238 FUNCTB = 0
0038 239
00000038 240 .PSECT $$$115_DRIVER
0038 241
0038 242 ;
0038 243 ; THE ASSOCIATED CLASS DRIVER USES THIS TABLE TO COMMAND THE PORT DRIVER.
0038 244 ; THE ADDRESS OF THIS TABLE IS CONTAINED IN THE TERMINAL UCB EXTENSION AREA.
0038 245 ; THE OFFSET DEFINITIONS ARE DEFINED BY TTYDEFS.
0038 246 ;
0038 247
0038 248 PORT_VECTOR:
0038 249
```

```
0038 250 : DZ-11 SPECIFIC DISPATCH TABLE
0038 251 :
0038 252 :
0038 253 $VECINI DZ11,DZ$NULL
0070 254 $VEC STARTIO,DZ11$STARTIO
003C 255 $VEC SET LINE,DZ$SET_LINE
0044 256 $VEC DS SET,DZ11$DS_SET
0048 257 $VEC XON,DZ11$XON
004C 258 $VEC XOFF,DZ11$XOFF
0050 259 $VEC STOP,DZ$STOP
0054 260 $VEC ABORT,DZ$ABORT
005C 261 $VEC RESUME,DZ11$RESUME
0060 262 $VEC SET MODEM,DZ11$SET_MODEM
0064 263 $VEC MAINT,DZ11$MAINT
006C 264 .IF NDF DZV
006C 265 $VECEND END=NO
0070 266 :
0070 267 : DZ-32 SPECIFIC DISPATCH TABLE
0070 268 :
0070 269 :
0070 270 $VECINI DZ32,DZ$NULL
00A8 271 $VEC STARTIO,DZ32$STARTIO ; START NEW OUTPUT
0074 272 $VEC SET LINE,DZ$SET_LINE ; SET NEW PARITY/SPEED
007C 273 $VEC DS SET,DZ32$DS_SET ; SET NEW OUTPUT MODEM SIGNALS
0080 274 $VEC XON,DZ32$XON ; SEND XON
0084 275 $VEC XOFF,DZ32$XOFF ; SEND XOFF
0088 276 $VEC STOP,DZ$STOP ; STOP CURRENT OUTPUT
008C 277 $VEC ABORT,DZ$ABORT ; ABORT CURRENT OUTPUT
0094 278 $VEC RESUME,DZ32$RESUME ; RESUME STOPPED OUTPUT
0098 279 $VEC MAINT,DZ32$MAINT ; INVOKE MAINTENANCE FUNCTIONS
00A4 280 .ENDC
00A4 281 $VECEND
00AC 282 DZ$NULL: ; NULL PORT ROUTINE
05 00AC 283 RSB
00AD 284
00AD 285
```

```
.SBTTL REGISTER DEFINITIONS

00AD 287
00AD 288
00AD 289
00AD 290 : CSR BIT DEFINITIONS ( CSR ) ( READ/WRITE )
00AD 291 :
00AD 292 $VIELD DZCSR,0,<-
00AD 293 <MODE,1,M>,- : DZ32 - MODE/ DZ11 - UNUSED
00AD 294 <DS_ENAB,1,M>,- : DZ32 - DATA SET INTERRUPT ENABLE
00AD 295 <1>,- : UNUSED
00AD 296 <MAINT,1,M>,- : LINE TURNAROUND
00AD 297 <CLEAR,1,M>,- : MASTER RESET
00AD 298 <MASTENAB,1,M>,- : MASTER SCAN ENABLE
00AD 299 <RCVINT,1,M>,- : RECEIVER INTERRUPT ENABLE
00AD 300 <RCVRDY,1,M>,- : RECEIVER READY
00AD 301 <LINE,3,M>,- : LINE NUMBAE ( 0 - 7 )
00AD 302 <DS_CHG,1,M>,- : DZ32 - DATA SET INTERRUPT
00AD 303 <2>,- : UNUSED
00AD 304 <SNDINT,1,M>,- : TRANSMIT INTERRUPT ENABLE
00AD 305 <SNDRDY,1,M>,- : TRANSMITTER READY
00AD 306 >
00AD 307 :
00AD 308 : RECEIVER BUFFER ( CSR+2 ) ( READ ONLY )
00AD 309 :
00AD 310 $VIELD DZRCV,0,<-
00AD 311 <BUF,8,M>,- : RECEIVER DATA
00AD 312 <LINE,3,M>,- : LINE NUMBER ( 0 - 7 )
00AD 313 <1>,- :
00AD 314 <PARERR,1,M>,- : PARITY ERROR
00AD 315 <FRAMER,1,M>,- : FRAME ERROR
00AD 316 <OVERRUN,1,M>,- : OVERRUN ERROR
00AD 317 <VALID,1,M>,- : DATA VALID
00AD 318 >
00AD 319 :
00AD 320 : LINE PARAMETER REGISTER ( CSR+2 ) ( WRITE ONLY )
00AD 321 :
00AD 322 :
00AD 323 $VIELD DZLPR,0,<-
00AD 324 <LINE,3,M>,- : LINE NUMBER (0-7)
00AD 325 <SIZE,2,M>,- : CHARACTER SIZE
00AD 326 <STOP,1,M>,- : NUMBER STOP BITS
00AD 327 <PARITY,1,M>,- : PARITY ENABLE
00AD 328 <ODD,1,M>,- : ODD PARITY
00AD 329 <SPEED,4,M>,- : LINE SPEED
00AD 330 <CLOCK,1,M>,- : RECEIVER CLOCK
00AD 331 <SPLIT,1,M>,- : DZ32 - SPLIT SPEED
00AD 332 >
00AD 333 :
00AD 334 : DZ-32 SPECIFIC MODEM CONTROL
00AD 335 :
00AD 336 :
00AD 337 :
00AD 338 $VIELD DZLCS1,8,<-
00AD 339 <7>,- :
00AD 340 <ACK,1,M>,- : READY FOR COMMAND/ UPDATE OUTPUT MODEM
00AD 341 >
00AD 342
00AD 343
```

DZDRIVER
V04-000

- Port Driver for DZ-11 support L 12
REGISTER DEFINITIONS

16-SEP-1984 02:24:50 VAX/VMS Macro V04-00
5-SEP-1984 04:15:55 [TTDRVR.SRC]DZDRIVER.MAR;1

Page 8
(1)

00AD 344

```
00AD 346 .SBTTL CONTROLLER INITIALIZATION
00AD 347
00AD 348
00AD 349
00AD 350
00AD 351 :++
00AD 352 : DZ$INITIAL - INITIALIZE INTERFACE
00AD 353 :
00AD 354 : FUNCTIONAL DESCRIPTION:
00AD 355 : THIS ROUTINE IS ENTERED AT SYSTEM STARTUP AND POWER RECOVERY.
00AD 356 :
00AD 357 : INPUTS:
00AD 358 :
00AD 359 : R4 = ADDRESS OF THE UNIT CSR
00AD 360 : R5 = IDB OF UNIT
00AD 361 : R8 = ADDRESS OF THE UNIT CRB
00AD 362 :
00AD 363 : OUTPUTS:
00AD 364 :
00AD 365 : R2 is destroyed.
00AD 366 :
00AD 367 : IMPLICIT INPUTS:
00AD 368 :
00AD 369 : IPL = IPL$_POWER
00AD 370 :
00AD 371 :--
00AD 372 DZ$INITIAL:: : INITIALIZE DZ UNIT
00AD 373 :
00AD 374 :
00AD 375 : SET UP CONTROLLER
00AD 376 :
00AD 377 : class_ctrl_init dz$dpt,port_vector
00AD 378 :
64 10 B0 00DA 379 25$: MOVW #DZCSR$_CLEAR,(R4) : INIT CONTROLLER RESET
00DD 380 :
00DD 381 :
00DD 382 : WAIT TILL CONTROLLER INITIALIZATION IS COMPLETE
00DD 383 :
00DD 384 : TIMEWAIT #500,#DZCSR$_CLEAR,(R4),W,.FALSE.
0104 385 :
64 4063 8F B0 0104 386 MOVW #<<DZCSR$_MASTENAB>!-- :
0109 387 : <DZCSR$_RCVINT>!-- : ENABLE RECEIVER INTERRUPTS
0109 388 : <DZCSR$_SNDINT>!-- : ENABLE TRANSMITTER INTERRUPTS
0109 389 : <DZCSR$_DS_ENAB>!-- : ENABLE DZ-32 DATA SET INTERRUPTS
0109 390 : <DZCSR$_MODE>>,(R4) : ENABLE ENHANCED MODE ON DZ-32
3E 50 E9 0109 391 BLBC R0,DZ$CTRL_ERROR
010C 392 :
19 52 64 B0 010C 393 MOVW (R4),R2 : GET NEW STATUS
19 52 00 E0 010F 394 BBS #DZCSR$_V_MODE,R2,110$ : BRANCH IF DZ-32 CONTROLLER
0113 395 :
0113 396 100$: MOVB #DZ$_DZ11,CRB$_TT_TYPE(R8); CONTROLLER IS DZ11
0B A8 42 8F 90 0113 397 :
0118 398 :
0118 399 : INIT DZ-11 INTERRUPT VECTORS
0118 400 : THIS IS DONE HERE TO ALLOW THE DRIVER TO SERVICE INTERUPTS
0118 401 : FOR BOTH THE DZ-11 AND DZ-32 BETWEEN CONTROLLER AN UNIT INIT.
0118 402 :
```

```
28 A8 000003E9'EF DE 0118 403          MOVAL DZ11$INTINP,CB$SL_INTD+4(R8) ; INIT RECEIVER VECTOR
4C A8 000006E9'EF DE 0120 404          MOVAL DZ11$INTOUT,CB$SL_INTD2+4(R8) ; INIT TRANSMITTER VECTOR
                                0128 405
                                1C A8 B4 0128 406          CLRW CB$B_DZ_RING(R8) ; RESET CURRENT DZ-11 MODEM STATE
                                05 012B 407          RSB
                                012C 408
                                012C 409 110$: :DZ-32 CONTROLLER SPECIFIC INIT
                                012C 410 .if ndf DZV
                                012C 411
                                0B A8 43 8F 90 012C 412          MOVB #DTS DZ32,CB$B_TT_TYPE(R8) ; CONTROLLER IS DZ-32
                                0E A5 94 0131 413          CLRB IDB$B_TT_ENABLE(R5) ; RESET DZ-32 LINE ENABLE
                                07 A4 0E A5 90 0134 414          MOVB IDB$B_TT_ENABLE(R5),7(R4) ; RESET TRANSMIT LINE ENABLES
                                0139 415
                                : : INIT DZ-32 ALTERNATE INTERRUPT VECTORS
                                0139 416
                                0139 417
                                28 A8 00000477'EF DE 0139 418          MOVAL DZ32$INTINP,CB$SL_INTD+4(R8) ; INIT RECEIVER VECTOR
4C A8 000007A3'EF DE 0141 419          MOVAL DZ32$INTOUT,CB$SL_INTD2+4(R8) ; INIT TRANSMITTER VECTOR
                                0149 420 .endc
                                05 0149 421          RSB
                                014A 422
                                014A 423 DZ$CTRL_ERROR:
                                05 014A 424          RSB
                                014B 425
```

```
014B 427 .SBTTL UNIT INITIALIZATION
014B 428 .++
014B 429 DZ$INITLINE - UNIT INITIALIZATION
014B 430
014B 431 FUNCTIONAL DESCRIPTION:
014B 432
014B 433 THIS ROUTINE PERFORMS A SIMPLE UNIT INITIALIZATION.
014B 434
014B 435 INPUTS:
014B 436
014B 437 R5 = UCB ADDRESS
014B 438
014B 439 OUTPUTS:
014B 440
014B 441 R2,R5 ARE PRESERVED.
014B 442
014B 443
014B 444 DZ$INITLINE::
014B 445
014B 446 MOVL UCB$L_CRB(R5),R4 ; GET CRB ADDRESS
014F 447
014F 448 .IF NDF DZV ; IF NOT DZV
014F 449
014F 450 MOVAL DZ32$VEC,R0 ; SET DZ-32 PORT VECTOR TABLE
0154 451 CMPB #DTS_DZ32,CRB$B_TT_TYPE(R4) ; IS IT DZ-32 ?
0159 452 BEQL 5$ ; YES
015B 453
015B 454 .ENDC ; END OF DZ32 CODE
015B 455
015B 456 MOVAL DZ11$VEC,R0 ; SET DZ-11 PORT VECTOR TABLE
0160 457 5$: CLASS_UNIT_INIT
01A9 458 BISW #UCB$M_ONLINE,UCB$W_STS(R5); SET ONLINE
01AD 459 10$: ASHL UCB$W_UNIT(R5),#1,R3 ; BUILD UNIT'S BIT MASK
01B2 460 MOVW R3,UCB$W_TT_UNITBIT(R5) ; SAVE IT
01B7 461 MOVL UCB$L_TT_CLASS(R5),R1 ; ADDRESS CLASS VECTOR TABLE
01BC 462 JSB @CLASS_SETUP_UCB(R1) ; INIT UCB FIELDS
01BF 463 20$:
01BF 464 JSB DZ$SET_LINE ; INIT SPEED/PARITY
01C5 465
01C5 466 MOVL UCB$L_CRB(R5),R4 ; GET CRB ADDRESS
01C9 467
01C9 468 .IF NDF DZV ; IF NOT DZV
01C9 469
01C9 470 CMPB #DTS_DZ11,CRB$B_TT_TYPE(R4) ; CONTROLLER DZ11?
01CE 471 BEQL 25$ ; YES
01D0 472
01D0 473
01D0 474
01D0 475 INIT RECEIVER MODEM STATUS FOR DZ-32
01D0 476
01D0 477 MOVL @CRB$L_INTD+VEC$L_IDB(R4),R4 ; GET CSR ADDRESS
01D4 478
01D4 479 ; WAIT TILL MODEM CONTROL READY FOR COMMAD
01D4 480
01D4 481 TIMEWAIT #500,#DZLCS1$M_ACK,4(R4),W,.TRUE.
01FE 482 BLBC R0,DZ$UNIT_ERROR
0201 483 MOVW UCB$W_UNIT(R5),4(R4) ; REQUEST STATUS ON LINE
```

```
0124 C5 24 50 E9 01206 484
04 A4 90 0206 485 ; WAIT FOR COMPLETION
0206 486
0206 487 TIMEWAIT #500,#DZLCS1$M_ACK,4(R4),W,.TRUE.
0230 488 BLBC R0,DZ$UNIT_ERROR
0233 489 MOVW 4(R4),UCB$B_TT_DS_RCV(R5) ; UPDATE RECEIVER MODEM STATUS
0239 490
0239 491 25$:
0239 492 .ENDC ; END OF DZ32 CODE
0239 493
0239 494 PUSHL R2
50 51 00 9A 023B 495 MOVZBL #MODEM$C_INIT,R1 ; ASSUME INIT MODEM PROTOCOL
0114 C5 D0 023E 496 MOVL UCB$L_TT_CLASS(R5),R0 ; ADDRESS CLASS VECTOR TABLE
0C B0 16 0243 497 JSB @CLASS_DS_TRAN(R0) ; INVOKE TO INIT MODEM PROTOCOL
52 8ED0 0246 498 POPL R2
08 64 A5 05 E1 0249 499 30$:
50 0114 C5 D0 024E 500 BBC #UCB$V_POWER,UCB$W_STS(R5),40$; DID WE DETECT A POWER FAIL
20 B0 17 0253 501 MOVL UCB$L_TT_CLASS(R5),R0 ; GET THE CLASS VECTOR TABLE ADDRESS
05 05 0256 502 JMP @CLASS_POWERFAIL(R0) ; AND GOTO THE POWERFAIL CODE
0257 503 40$:
0257 504 RSB
0257 505 :: ERROR DETECTED DURING INITIALIZATION
0257 506 ::
0257 507
0257 508 DZ$UNIT_ERROR:
64 A5 10 AA 0257 509 BICW #UCB$M_ONLINE,UCB$W_STS(R5) ; UNIT NOT ON LINE
05 025B 510 RSB
```

```
025C 512 .SBTTL MAINTENANCE ROUTINES
025C 513
025C 514 ++
025C 515 DZ$MAINT - MAINTENANCE FUNCTIONS
025C 516
025C 517 FUNCTIONAL DESCRIPTION:
025C 518
025C 519 THIS ROUTINE PERFORMS MAINTENANCE FUNCTIONS FOR THE DZ .
025C 520
025C 521 INPUTS:
025C 522
025C 523 R5 = UBC ADDRESS
025C 524 UCB$B_TT_MAINT = FUNCTION TO BE PERFORMED
025C 525
025C 526 OUTPUTS:
025C 527
025C 528 --
025C 529 .IF NDF DZV
025C 530 DZ32$MAINT:
025C 531 BITB #IOSM_LOOP2-7,- ; LOOPBACK FUNCTION
025E 532 UCB$B_TT_MAINT(R5)
0261 533 BEQL 5$ ; NO
52 40 8F 9A 0263 534 MOVZBL #^X40,R2 ; SPECIFY LOOPBACK CODE
0C 11 0267 535 BRB 10$
0269 536 5$:
0269 537 BITB #IOSM_UNLOOP2-7,- ; RESET LOOPBACK FUNCTION
026B 538 UCB$B_TT_MAINT(R5)
026E 539 BEQL 15$ ; NO
52 7200 8F 3C 0270 540 MOVZWL #^X7200,R2 ; SPECIFY UNLOOP CODE (BOTH)
000002ED'EF 16 0275 541 10$: JSB DZ32$DS_SET ; UPDATE CONTROLLER
50 01 9A 027B 542 MOVZBL #1,R0 ; INDICATE SUCCESS
05 027E 543
027F 544 RSB
027F 545 15$:
027F 546 BITB #IOSM_LOOP_EXT2-7,- ; LOOPBACK FUNCTION
0281 547 UCB$B_TT_MAINT(R5)
0284 548 BEQL 20$ ; NO
52 72 8F 9A 0286 549 MOVZBL #^X72,R2 ; SPECIFY LOOPBACK CODE
E9 11 028A 550 BRB 10$
028C 551
028C 552 20$:
028C 553 .ENDC
028C 554 DZ11$MAINT:
028C 555 BITB #IOSM_LINE_OFF2-7,- ; LINE OFF
028E 556 UCB$B_TT_MAINT(R5)
0291 557 BEQL 10$ ; NO
80 8F 88 0293 558 BISB #UCB$M_TT_DSBL,- ; DISABLE LINE
012A C5 0296 559 UCB$B_TT_MAINT(R5)
0D 11 0299 560 BRB 20$
029B 561 10$:
029B 562 BITB #IOSM_LINE_ON2-7,- ; LINE ON
029D 563 UCB$B_TT_MAINT(R5)
02A0 564 BEQL 30$ ; NO
80 8F 8A 02A2 565 BICB #UCB$M_TT_DSBL,- ; REENABLE LINE
012A C5 02A5 566 UCB$B_TT_MAINT(R5)
00000864'EF 16 02A8 567 20$: JSB DZ$SET_LINE ; IMPLEMENT FUNCTION
02A8 568
```

DZDRIVER
V04-000

- Port Driver for DZ-11 support E 13
MAINTENANCE ROUTINES

16-SEP-1984 02:24:50 VAX/VMS Macro V04-00
5-SEP-1984 04:15:55 [TTDRVR.SRC]DZDRIVER.MAR;1

Page 14
(1)

50	01	9A	02AE	569		MOVZBL	#1,R0
		05	02B1	570		RSB	
			02B2	571	30\$:		
	50	D4	02B2	572		CLRL	R0
		05	02B4	573		RSB	
			02B5	574			
			02B5	575			

DZ
V0

```
02B5 577 .SBTTL OUTPUT MODEM CONTROL
02B5 578 :++
02B5 579 DZ$DS_SET - SET OUTPUT MODEM SIGNALS
02B5 580 :
02B5 581 FUNCTIONAL DESCRIPTION:
02B5 582 :
02B5 583 THIS ROUTINE OUTPUTS THE OUTPUT MODEM SIGNALS FOR THE SPECIFIED UNIT
02B5 584 :
02B5 585 INPUTS:
02B5 586 :
02B5 587 R2 = LOW BYTE - SIGNALS TO ACTIVATE
02B5 588 HIGH BYTE- SIGNALS TO DEACTIVATE
02B5 589 :
02B5 590 R5 = UBC ADDRESS
02B5 591 :
02B5 592 OUTPUTS:
02B5 593 :
02B5 594 R0-R3 ARE USED.
02B5 595 :--
02B5 596 :
02B5 597 DZ11$DS_SET:
02B5 598 PUSHL R4 ; SAVE
02B7 599 BISB R2,UCB$B_TT_DS_TX(R5) ; SET NEW OUTPUT SIGNALS
52 0125 C5 52 88 02B7 600 ASHL #8,R2,R2 ; ACCESS SIGNALS TO RESET
52 52 F8 8F 78 02BC 601 BICB R2,UCB$B_TT_DS_TX(R5) ; RESET THEM
0125 C5 52 8A 02C1 602 MOVL UCB$C_CRB(R5),R4 ; GET CRB ADDRESS
54 24 A5 D0 02C6 603 MOVL @CRB$C_INTD+VEC$C_IDB(R4),R3 ; GET CSR ADDRESS
53 2C B4 D0 02CA 604 EXTZV #TT$V_DS_DTR,#1,UCB$B_TT_DS_TX(R5),R1 ; GET CURRENT DTR FOR LINE
51 0125 C5 01 01 EF 02CE 605 ASHL UCB$W_UNIT(R5),R1,R1 ; SHIFT TO RELATIVE LINE POSITION
51 51 54 A5 78 02D5 607 BICB UCB$W_TT_UNITBIT(R5),CRB$B_DZ_DTR(R4) ; RESET CURRENT DTR FOR THAT LINE
1E A4 0106 C5 8A 02DA 608 BISB R1,CRB$B_DZ_DTR(R4) ; SET IT IF NEED BE
05 A3 1E A4 90 02E4 611 MOV B CRB$B_DZ_DTR(R4),5(R3) ; UPDATE DTR STATUS FOR LINES
54 8ED0 02E9 612 POPL R4
05 02EC 613 RSB
02ED 614 .IF NDF DZV
02ED 615 DZ32$DS_SET:
02ED 616 PUSHL R4 ; SAVE
52 0125 C5 52 88 02EF 617 BISB R2,UCB$B_TT_DS_TX(R5) ; SET NEW OUTPUT SIGNALS
52 52 F8 8F 78 02F4 618 ASHL #8,R2,R2 ; ACCESS SIGNALS TO RESET
0125 C5 52 8A 02F9 619 BICB R2,UCB$B_TT_DS_TX(R5) ; RESET THEM
54 24 A5 D0 02FE 620 MOVL UCB$C_CRB(R5),R4 ; GET CRB ADDRESS
53 2C B4 D0 0302 621 MOVL @CRB$C_INTD+VEC$C_IDB(R4),R3 ; WAIT FOR READY
7E 0124 C5 3C 0306 622 TIMEWAIT #100,#DZLCS1$M_ACK,4(R3),W,.TRUE. ; CREATE TEMP LOCATION
6E 54 A5 90 0330 623 MOVZWL UCB$B_TT_DS_TX-T(R5),-(SP)
6E 8000 8F A8 0335 624 MOV B UCB$W_UNIT(R5), (SP) ; SET UNIT NUMBER
04 A3 8E F7 0339 625 BISW #DZLCS1$M_ACK,(SP) ; ENABLE NEW OUTPUT SIGNALS
54 8ED0 033E 626 CVTLW (SP)+,4(R3) ; SET NEW OUTPUT MODEM SIGNALS
05 0342 627 POPL R4
0343 628 RSB
0346 629 .ENDC
```

```
0346 631 .SBTTL DZ-11 MODEM POLLER
0346 632 :++
0346 633 :DZ$TIMER - POLL FOR DZ-11 MODEM TRANSITIONS
0346 634 :
0346 635 :FUNCTIONAL DESCRIPTION:
0346 636 :
0346 637 :THIS ROUTINE CHECKS FOR DZ-11 CONTROLLER MODEM
0346 638 :TRANSITION. IT UPDATES THE INPUT MODEM STATUS FOR EACH
0346 639 :LINE AND CALLS THE CLASS TRANSITION ROUTINE FOR EACH LINE WITH
0346 640 :A CHANGE.
0346 641 :
0346 642 :INPUTS:
0346 643 :
0346 644 :R5 - TQE ADDRESS
0346 645 :
0346 646 :OUTPUTS:
0346 647 :
0346 648 :R0 - R4 DESTROYED
0346 649 :
0346 650 :--
0346 651 :
0346 652 :DZ$TIMER:
54 0060 8F BB 0346 653 :PUSHR #^M<R5,R6>
00000000'EF DE 034A 654 :MOVAL DZ$L_DIALUP,R4 ; GET DZ TIMER LIST HEAD
54 64 DO 0351 655 5$:
05 12 0351 656 :MOVL (R4),R4 ; GET NEXT CRB ADDRESS
0060 8F BA 0354 657 :BNEQ 15$ ; PROCESS LINES FOR THIS CRB
05 05 0356 658 :POPR #^M<R5,R6> ; RESTORE REGISTERS
05 05 035A 659 :RSB ; RETURN FROM TIMER INTERRUPT
035B 660 :
035B 661 :TEST LINES ON THIS CONTROLLER FOR A TRANSITION
035B 662 :
035B 663 :
035B 664 15$:
54 54 DD 035B 665 :PUSHL R4 ; SAVE TIMER THREAD
53 18 C2 035D 666 :SUBL #CRB$L_DZ_MODEM,R4 ; GET ACTUAL CRB ADDRESS
06 A3 2C B4 DO 0360 667 :MOVL @CRB$L_INTD+VEC$L_IDB(R4),R3 ; GET CSR ADDRESS
1C A4 B1 0364 668 :CMPW CRB$B_DZ_RING(R4),6(R3) ; ANY TRANSITIONS
78 13 0369 669 :BEQL 60$ ; NONE
036B 670 :
036B 671 :FIND WHICH SIGNALS CHANGED AND UPDATE THEM
036B 672 :
50 52 06 A3 90 036B 673 :MOVB 6(R3),R2 ; GET NEW RING
1C A4 52 8D 036F 674 :XORB3 R2,CRB$B_DZ_RING(R4),R0 ; FIND TRANSITIONED LINES
1C A4 52 90 0374 675 :MOVB R2,CRB$B_DZ_RING(R4) ; UPDATE CURRENT RING
52 07 A3 90 0378 676 :MOVB 7(R3),R2 ; GET NEW CARRIER
56 1D A4 52 8D 037C 677 :XORB3 R2,CRB$B_DZ_CARRIER(R4),R6
1D 50 56 88 0381 678 :BISB R6,R0 ; FLAG LINES WITH TRANSITIONED CARRIER
1D A4 52 90 0384 679 :MOVB R2,CRB$B_DZ_CARRIER(R4) ; UPDATE CURRENT CARRIER
0388 680 :
0388 681 :PROCESS TRANSITIONED LINES
0388 682 :
0388 683 :
51 50 08 00 EA 0388 684 50$: FFS #0,#8,R0,R1 ; FIND NEXT LINE NEEDING SERVICE
54 13 038D 685 :BEQL 60$ ; DONE
00 50 51 E5 038F 686 :BBCC R1,R0,55$ ; RESET ATTENTION BIT FOR THIS LINE
0393 687 55$:
```

```

      56 2C A4 D0 0393 688
      55 18 A641 D0 0397 689
      E5 44 A5 EA 13 039C 690
      E1 039E 691
      03A3 692
      03A3 693
56 1C A4 01 51 EF 03AA 694
   01 06 56 FO 0380 695
      0124 C5 03B4 696
56 1D A4 01 51 EF 03B7 697
   01 05 56 FO 03BD 698
      0124 C5 03C1 699
      90 8F 88 03C4 700
      0124 C5 03C7 701
52 0124 C5 90 03CA 702
   51 03 9A 03CF 703
56 0114 C5 D0 03D2 704
   1F BB 03D7 705
      OC B6 16 03D9 706
      1F BA 03DC 707
      A5 11 03DE 708
      03E1 709
      03E3 710
      03E3 711 60$:
      54 8ED0 03E3 712
      FF68 31 03E6 713
      03E9 714
      03E9 715

      MOVL CRBSL_INTD+VECSL_IDB(R4),R6
      MOVL IDBSL_UCBLST(R6)[R1],R5 ; GET UCB FOR THAT LINE
      BEQL 50$ ; NONE
      BBC #TT$V_MODEM,UCBSL_DEVDEPEND(R5),50$ ; SKIP IF NOT MODEM LINE
      DSBINT UCBSB_DIPL(R5) ; RAISE TO DEVICE IPL
      EXTZV R1,#1-CRBSB_DZ_RING(R4),R6 ; GET RING FOR THAT LINE
      INSV R6,#TT$V_DS_RING,#1,- ; UPDATE IT IN UCB
      UCBSB_TT_DS_RCV(R5)
      EXTZV R1,#1-CRBSB_DZ_CARRIER(R4),R6 ; GET CD FOR THAT LINE
      INSV R6,#TT$V_DS_CARRIER,#1,- ; UPDATE IT IN UCB
      UCBSB_TT_DS_RCV(R5)
      BISB #<TT$M_DS_DSR!TT$M_DS_CTS>,- ; ASSUME CTS AND DSR ALWAYS SET
      UCBSB_TT_DS_RCV(R5)
      MOVB UCBSB_TT_DS_RCV(R5),R2 ; GET CURRENT RECV MODEM STATUS
      MOVZBL #MODEM$C_DATASET,R1 ; SIGNAL DATASET TRANSITION
      MOVL UCBSL_TT_CLASS(R5),R6 ; GET CLASS VECTOR TABLE
      PUSHF #^M<R0,R1,R2,R3,R4> ; SAVE VOLITAL REGISTERS
      JSB @CLASS_DS_TRAN(R6) ; SIGNAL TRANSITION
      POPR #^M<R0,R1,R2,R3,R4> ; RESTORE REGISTERS
      ENBINT ; RESTORE IPL
      BRB 50$

      POPL R4 ; RESTORE TIMER THREAD
      BRW 5$
```

```
03E9 717 .SBTTL RECEIVER INTERRUPT SERVICE
03E9 718 :++
03E9 719 :DZ$INTINP - DZ RECEIVER READY INTERRUPTS
03E9 720 :
03E9 721 :FUNCTIONAL DESCRIPTION:
03E9 722 :
03E9 723 :THIS ROUTINE IS ENTERED WHEN A CHARACTER IS AVAILABLE IN THE UNIT'S
03E9 724 :SILO. THE CHARACTER IS EXTRACTED AND IS PASSED TO THE ASSOCIATED
03E9 725 :CLASS DRIVER. IF THE CLASS DRIVER RETURNS CHARACTER(S) THEN NEW
03E9 726 :OUTPUT IT INITIATED (NORMALLY ECHO).
03E9 727 :
03E9 728 :INPUTS:
03E9 729 :
03E9 730 :OO(SP) = ADDRESS OF IDB
03E9 731 :
03E9 732 :IMPLICIT INPUTS:
03E9 733 :
03E9 734 :R0,R1,R2,R3,R4,R5 ARE SAVED ON STACK.
03E9 735 :
03E9 736 :OUTPUTS:
03E9 737 :
03E9 738 :THE INTERRUPT IS DISMISSED WHEN THE SILO IS EMPTY.
03E9 739 :
03E9 740 :--
03E9 741 :DZ11$INTINP:: : DZ-11 INPUT INTERRUPTS
03E9 742 :
03E9 743 :GET THE CSR ADDRESS
03E9 744 :
54 9E D0 03E9 745 :MOVL @ (SP)+,R4 : GET THE IDB ADDRESS
50 54 DD 03EC 746 :PUSHL R4 : SAVE IDB ADDRESS
50 64 D0 03EE 747 :MOVL (R4),R0 : GET THE CSR ADDRESS
03F1 748 :
03F1 749 :GET THE CHARACTER FROM THE INTERFACE
03F1 750 :
53 02 A0 B0 03F1 751 25$: MOVW 2(R0),R3 : GET THE CHARACTER, ERRORS AND LINE NUMBER
53 7000 8F B3 03F5 752 :BGEQ 100$ : SILO EMPTY
03F7 753 :BITW #<DZRCV$M_PARERR>,-
03FC 754 :<DZRCV$M_OVERRUN>,-
03FC 755 :<DZRCV$M_FRAMER>,R3 : ERRORS?
03FC 756 :BNEQ 50$ : YES, PROCESS THEM
52 53 F8 8F 78 03FE 757 27$: ASHL # -8, R3, R2 : GET THE LINE NUMBER
52 FFFFFFFF 8F CA 0403 758 :BICL #^C<7>, R2
53 53 53 9A 040A 759 :MOVZBL R3, R3 : CLEAR THE HIGH BYTES OF CHARACTER
55 18 A442 D0 040D 760 :MOVL IDB$$_UCBLST(R4)[R2], R5 : GET THE UCB FOR THAT LINE
0110 D5 16 0412 761 :BEQL 25$ : IF EQL THEN NOT THERE
0108 C5 95 0418 762 :JSB @UCB$$_TT_PUTNXT(R5) : BUFFER THE CHARACTER
0108 C5 17 041C 763 :TSTB UCB$$_TT_OUTYPE(R5) : DID HE RETURN ANYTHING TO OUTPUT
0108 C5 53 90 041E 764 :BLEQ 40$ : NONE OR STRING OUTPUT
0400 8F AB 0423 765 :MOVB R3, UCB$$_TT_HOLD(R5) : SAVE THE CHARACTER IN TANK
0108 C5 AB 0427 766 :BISW #TTYSM_TANK_HOLD,- : SIGNAL CHARACTER IN TANK
04 A0 0106 C5 AB 042A 767 :UCB$$_TT_HOLD(R5)
04 A0 54 6E D0 0430 768 30$: BISW UCB$$_TT_UNITBIT(R5), 4(R0) : ENABLE LINE
54 BC 11 0433 769 :MOVL (SP), R4 : GET IDB ADDRESS
0435 770 :BRB 25$ : CONTINUE
0435 771 40$: BEQL 30$ : NO CHARACTER
0800 8F AB 0437 772 :BISW #TTYSM_TANK_BURST,- : SIGNAL BURST
0437 773
```

```
04 A0 0108 C5 043B 774 UCBSW TT_HOLD(R5)
0106 C5 A8 043E 775 BISW UCBSW TT_UNITBIT(R5),4(R0) ; ENABLE LINE
EA 11 0444 776 BRB 30$
0446 777 :
0446 778 : SILO EMPTY OR CHARACTER IN ERROR
0446 779 :
0446 780 50$:
0446 781 :
0446 782 : PROCESS PARITY, FRAME OR OVERRUN ERROR
0446 783 :
52 53 F8 8F 78 0446 784 ASHL #-8,R3,R2 ; GET LINE NUMBER
52 FFFFFFFF 8F CA 044B 785 BICL #^C<7>,R2 ;
55 18 A442 D0 0452 786 MOVL IDB$L_UCBLST(R4)[R2],R5 ; GET UCB ADDRESS
0A 13 0457 787 BEQL 70$ ; IF EQL THEN NO UCB
52 0114 C5 D0 0459 788 MOVL UCB$L_TT_CLASS(R5),R2 ; GET CLASS DISPATCH
045E 789
14 B2 16 045E 790 60$: JSB @CLASS_READERROR(R2) ; SIGNAL ERROR
9B 12 0461 791 BNEQ 27$ ; BRANCH WITH CHARACTER TO MAIN PATH
60 0080 8F B3 0463 792 70$: BITW #^X080,(R0) ; VALID CHARACTER IN SILO NOW?
C6 12 0468 793 BNEQ 30$ ; IF NEQ THEN YES
5E 04 C0 046A 794 100$: ADDL #4,SP ; REMOVE IDB ADDRESS
50 8E 7D 046D 795 MOVQ (SP)+,R0 ; RESTORE REGISTERS
52 8E 7D 0470 796 MOVQ (SP)+,R2 ;
54 8E 7D 0473 797 MOVQ (SP)+,R4 ;
02 0476 798 REI ;
0477 799
```

```
0477 801 .IF NDF DZV
0477 802 :
0477 803 : DZ-32 INPUT INTERRUPT SERVICE
0477 804 :
0477 805 :
0477 806 DZ32$INTINP:: ; DZ-32 INPUT INTERRUPTS
0477 807 :
0477 808 : GET THE CSR ADDRESS
0477 809 :
54 9E D0 0477 810 MOVL 2(SP)+,R4 ; GET THE IDB ADDRESS
54 DD 047A 811 PUSHL R4 ; SAVE IDB ADDRESS
50 64 D0 047C 812 MOVL (R4),R0 ; GET THE CSR ADDRESS
AA 047F 813 BICW #<<DZCSR$M_RCVINT>!-- ; DISABLE RECEIVER INTERRUPTS
0480 814 <DZCSR$M_DS_ENAB>!-- ; DISABLE DZ-32 DATA SET INTERRUPTS
60 0042 8F 0480 815 (R0) ; DZ-32 during the interrupt service routine
0484 816 :
0484 817 : GET THE CHARACTER FROM THE INTERFACE
0484 818 :
53 02 A0 B0 0484 819 25$: MOVW 2(R0),R3 ; GET THE CHARACTER, ERRORS AND LINE NUMBER
53 7000 8F B3 0488 820 BGEQ 100$ ; SILO EMPTY
048A 821 BITW #<DZRCV$M_PARERR>!-- ; ERRORS?
048F 822 <DZRCV$M_OVERRUN>!-- ; ERRORS?
048F 823 <DZRCV$M_FRAMER>,R3 ; YES, PROCESS THEM
52 53 F8 4A 12 048F 824 BNEQ 50$ ; GET THE LINE NUMBER
52 FFFFFFFF 8F CA 0491 825 27$: ASHL #-8,R3,R2
53 53 53 9A 0496 826 BICL #^C<7>,R2
55 18 A442 D0 049D 827 MOVZBL R3,R3 ; CLEAR THE HIGH BYTES OF CHARACTER
0110 D5 16 04A0 828 MOVL IDB$L_UCBLST(R4)[R2],R5 ; GET THE UCB FOR THAT LINE
0108 C5 95 04A5 829 BEQL 25$ ; IF EQL THEN NOT THERE
0108 C5 1F 15 04A7 830 JSB 2UCB$L_TT_PUTNXT(R5) ; BUFFER THE CHARACTER
0400 8F A8 04AB 831 TSTB UCB$B_TT_OUTYPE(R5) ; DID HE RETURN ANYTHING TO OUTPUT
0108 C5 90 04AF 832 BLEQ 40$ ; NONE OR STRING OUTPUT
54 6E D0 04B1 833 MOVW R3,UCB$W_TT_HOLD(R5) ; SAVE THE CHARACTER IN TANK
0106 C5 88 04B6 834 BISW #TTYSM_TANK_HOLD,- ; SIGNAL CHARACTER IN TANK
0E A4 90 04BA 835 UCB$W_TT_HOLD(R5)
07 A0 0E A4 90 04BD 836 28$: MOVL (SP),R4 ; RESTORE IDB ADDRESS
54 6E D0 04BD 837 BISB UCB$W_TT_UNITBIT(R5),- ; ENABLE LINE
0106 C5 88 04C0 838 IDB$B_TT_ENABLE(R4)
0E A4 90 04C4 839 MOVB IDB$B_TT_ENABLE(R4),7(R0)
54 6E D0 04CB 842 30$: MOVL (SP),R4 ; GET IDB ADDRESS
B4 11 04CE 843 BRB 25$ ; CONTINUE
04D0 844 40$: BEQL 30$ ; NO CHARACTER
0800 8F A8 04D2 845 BISW #TTYSM_TANK_BURST,- ; SIGNAL BURST
0108 C5 11 04D6 846 UCB$W_TT_HOLD(R5)
E2 11 04D9 847 BRB 28$
04DB 848 :
04DB 849 : SILO EMPTY OR CHARACTER IN ERROR
04DB 850 :
04DB 851 :
04DB 852 50$:
04DB 853 :
04DB 854 : PROCESS PARITY, FRAME OR OVERRUN ERROR
04DB 855 :
52 53 F8 8F 78 04DB 856 ASHL #-8,R3,R2 ; GET LINE NUMBER
52 FFFFFFFF 8F CA 04E0 857 BICL #^C<7>,R2 ;
```

```
55 18 A442 D0 04E7 858 MOVL IDB$L_UCBLST(R4)[R2],R5 ; GET UCB ADDRESS
      0A 13 04EC 859 BEQL 70$ ; IF EQL THEN NO UCB
52 0114 C5 D0 04EE 860 MOVL UCB$L_TT_CLASS(R5),R2 ; GET CLASS DISPATCH
      14 B2 16 04F3 861
      99 12 04F6 862 60$: JSB @CLASS_READERROR(R2) ; SIGNAL ERROR
60 0080 8F B3 04F8 863 BNEQ 27$ ; BRANCH WITH CHARACTER TO MAIN PATH
      CC 12 04FD 864 70$: BITW #^X080,(R0) ; VALID CHARACTER IN SILO NOW?
      53 60 B0 04FF 865 BNEQ 30$ ; IF NEQ THEN YES
      12 53 0B E0 0502 866 100$: MOVW (R0),R3 ; TEST FOR MODEM TRANSITION
      50 04 C0 050B 867 BBS #DZCSR$V_DS_CHG,R3,200$ ; BRANCH IF MODEM TRANSITION
      50 8E 7D 050E 868 BISW #<<DZCSR$M_RCVINT>>!,- ; ENABLE RECEIVER INTERRUPTS
      52 8E 7D 0511 869 <DZCSR$M_DS_ENAB>>,- ; ENABLE DZ-32 DATA SET INTERRUPTS
      54 8E 7D 0514 870 (R0) ; DZ-32 BEFORE EXITING
      02 0517 871 ADDL #4,SP ; REMOVE IDB ADDRESS
      50 DD 0518 872 REI ; RESTORE REGISTERS
      53 06 A0 90 051A 873
52 53 FFFFFFFF 8F CB 051E 874
      55 18 A442 D0 0526 875
      0124 C5 53 90 052D 876 200$: PUSHL R0 ; SAVE R0
      51 03 9A 0535 877 MOVW 6(R0),R3 ; GET NEW RECEIVE MODEM SIGNALS
      53 0114 C5 D0 0538 878 BICL3 #^C<7>,R3,R2 ; ISOLATE UNIT NUMBER
      0C B3 16 053D 879 MOVL IDB$L_UCBLST(R4)[R2],R5 ; GET ASSOCIATED UCB
      50 8ED0 0540 880 BEQL 110$ ; NONE
      FF85 31 0543 881 MOVW R3,UCB$B_TT_DS_RCV(R5) ; UPDATE INPUT MODEM SIGNALS
      53 0114 C5 D0 0538 882 MOVZBL R3,R2 ; LOAD ARGUMENT
      0C B3 16 053D 883 MOVZBL #MODEM$C_DATASET,R1 ; MODEM TRANSITION
      50 8ED0 0540 884 MOVL UCB$L_TT_CLASS(R5),R3 ; ACCESS CLASS VECTORS
      FF85 31 0543 885 JSB @CLASS_DS_TRAN(R3) ; SIGNAL TRANSITION
      53 0114 C5 D0 0538 886 POPL R0 ; RESTORE R0
      0C B3 16 053D 887 110$: BRW 30$ ; DISMISS INTERRUPT
      FF85 31 0543 888
      53 0114 C5 D0 0538 889 .ENDC
```

```
0546 891 .SBTTL START I/O ROUTINE
0546 892 :++
0546 893 DZ$STARTIO - START I/O OPERATION ON DZ
0546 894 :
0546 895 FUNCTIONAL DESCRIPTION:
0546 896 :
0546 897 THIS ROUTINE IS ENTERED FROM THE DEVICE INDEPENDENT TERMINAL STARTIO
0546 898 ROUTINE TO ENABLE OUTPUT INTERRUPTS ON AN IDLE DZ UNIT.
0546 899 :
0546 900 INPUTS:
0546 901 :
0546 902 R3 = CHARACTER AND CC = PLUS
0546 903 ADDRESS AND CC = NEGATIVE
0546 904 :
0546 905 R5 = UCB ADDRESS
0546 906 :
0546 907 OUTPUTS:
0546 908 :
0546 909 R5 = UCB ADDRESS
0546 910 :--
0546 911 DZ11$STARTIO:: : START I/O ON UNIT
0546 912 BGEQ 20$ : SINGLE CHARACTER
0546 913 BISW #TTY$M_TANK_BURST,- : SIGNAL BURST ACTIVE
0546 914 UCB$W_TT_HOLD(R5)
0546 915 10$: MOVL UCB$W_TT_HOLD(R5),R1 : GET CRB OF UNIT
0546 916 MOVL @CRB$C_INTD+VEC$W_IDB(R1),R1 : GET CSR
0546 917 BISW UCB$W_TT_UNITBIT(R5),4(R1) : ENABLE LINE
0546 918 RSB : RETURN TO CALLER
0546 919 20$:
0546 920 MOVB R3,UCB$W_TT_HOLD(R5) : SAVE OUTPUT CHARACTER
0546 921 BISW #TTY$M_TANK_HOLD,- : SIGNAL CHARACTER IN TANK
0546 922 UCB$W_TT_HOLD(R5)
0546 923 BRB 10$
0546 924 :
0546 925 .IF NDF DZV
0546 926 DZ32$STARTIO:: : START I/O ON UNIT
0546 927 BGEQ 20$ : SINGLE CHARACTER SPECIFIED
0546 928 BISW #TTY$M_TANK_BURST,- : SIGNAL BURST ACTIVE
0546 929 UCB$W_TT_HOLD(R5)
0546 930 10$: MOVL UCB$W_TT_HOLD(R5),R1 : GET CRB OF UNIT
0546 931 MOVL CRB$W_INTD+VEC$W_IDB(R1),R4 : GET IDB ADDRESS
0546 932 MOVL (R4),R1 : GET CSR ADDRESS
0546 933 BISB UCB$W_TT_UNITBIT(R5),IDB$B_TT_ENABLE(R4)
0546 934 MOVB IDB$B_TT_ENABLE(R4),7(R1)
0546 935 RSB : RETURN TO CALLER
0546 936 20$:
0546 937 MOVB R3,UCB$W_TT_HOLD(R5) : SAVE OUTPUT CHARACTER
0546 938 BISW #TTY$M_TANK_HOLD,- : SIGNAL CHARACTER IN TANK
0546 939 UCB$W_TT_HOLD(R5)
0546 940 BRB 10$
0546 941 .ENDC
```

04 A1 0106 C5 16 18 0546 911
0800 8F A8 0546 912
0108 C5 0546 913
51 24 A5 D0 0546 914
51 2C B1 D0 0546 915
0106 C5 A8 0546 916
05 0546 917
055D 918
055E 919
0108 C5 53 90 055E 920
0400 8F A8 055E 921
0108 C5 055E 922
E3 11 055E 923
056C 924
056C 925
056C 926
056C 927
0800 8F 1E 18 056C 928
0108 C5 A8 056C 929
51 24 A5 D0 056C 930
54 2C A1 D0 056C 931
51 64 D0 056C 932
0E A4 0106 C5 88 056C 933
07 A1 0E A4 90 056C 934
05 056C 935
058C 936
0108 C5 53 90 058C 937
0400 8F A8 058C 938
0108 C5 058C 939
DB 11 058C 940
059A 941

```
059A 943 .SBTTL PORT ROUTINES STOP,RESUME,XON,XOFF
059A 944 :++
059A 945 DZ$XOFF - SEND XOFF
059A 946 DZ$XON - SEND XON
059A 947 DZ$STOP - STOP OUTPUT
059A 948 DZ$ABORT - ABORT CURRENT OUTPUT
059A 949 DZ$RESUME - RESUME STOPPED OUTPUT
059A 950
059A 951 FUNCTIONAL DESCRIPTION:
059A 952
059A 953 THESE ROUTINES ARE USED BY THE THE TERMINAL CLASS DRIVER TO
059A 954 CONTROL OUTPUT ON THE PORT
059A 955
059A 956 INPUTS:
059A 957
059A 958 R5 = UCB ADDRESS
059A 959
059A 960 OUTPUTS:
059A 961
059A 962 R5 = UCB ADDRESS
059A 963 --
059A 964 .ENABLE LSB
059A 965
059A 966 SCHEDULE XOFF TO BE SEND
059A 967
059A 968 INPUTS:
059A 969 R3 - CHARACTER TO BE SENT AS FLOW CONTROL
059A 970
059A 971 DZ11$XOFF:
059A 972
059A 973 SCHEDULE XON TO BE SENT
059A 974
059A 975 DZ11$XON:
059A 976 BISW #TTY$M_TANK_PREMPT,UCB$W_TT_HOLD(R5) ; SCHEDULE XON
059A 977 MOVW R3,UCB$B_TT_PREMPT(R5) ; SAVE THE CHARACTER
059A 978
059A 979 BBS #UCB$V_INT,UCB$W_STS(R5),10$ ; IF OUTPUT ACTIVE,
059A 980 ; FINISHED
059A 981 PUSHL R1 ; SAVE A REGISTER
059A 982 MOVL UCB$L_CRB(R5),R1 ; ACCESS CRB ADDRESS
059A 983 MOVL @CRB$C_INTD+VEC$L_IDB(R1),R1 ; GET CSR ADDRESS
059A 984 BISW UCB$W_TT_UNITBIT(R5),4(R1) ; ENABLE LINE
059A 985 POPL R1
059A 986 BBSS #UCB$V_INT,UCB$W_STS(R5),10$ ; SHOW OUTPUT ACTIVE
059A 987 10$:
059A 988 RSB
059A 989 .DISABLE LSB
059A 990
059A 991 STOP PORT OUTPUT
059A 992
059A 993 DZ$STOP:
059A 994 BISW #TTY$M_TANK_STOP,- ; SCHEDULE STOP
059A 995 UCB$W_TT_HOLD(R5)
059A 996 RSB
059A 997
059A 998 ABORT ANY CURRENT PORT OUTPUT ACTIVITY
059A 999
```

010B C5 0100 BF AB 059A 976
010A C5 53 90 05A1 977
18 64 A5 01 E0 05A6 978
51 24 A5 DD 05AB 980
51 2C B1 DO 05AD 982
04 A1 0106 C5 AB 05B1 983
00 64 A5 01 8ED0 05B8 985
05 05C3 987
05 05C3 988
05C4 989
05C4 990
05C4 991
05C4 992
05C4 993
0200 BF AB 05C4 994
010B C5 05 05C8 995
05 05CB 996
05CC 997
05CC 998
05CC 999

```
0108 C5 0B E5 05CC 1000 DZ$ABORT:
00 05CC 1001 BBCC #TTY$V_TANK_BURST,UCB$W_TT_HOLD(R5),- ; RESET BURST ACTIVE
05D1 1002 10$
05D2 1003 TIMSET 1 ; SET A TIMEOUT
05E5 1004 ; IN CASE OUTPUT ACTIVE
05E5 1005 RSB
05E6 1006
05E6 1007
05E6 1008 ; RESUME PREVIOUSLY STOPPED PORT OUTPUT
05E6 1009
05E6 1010
05E6 1011 DZ11$RESUME:
0108 C5 0200 51 DD 05E6 1012 PUSHL R1 ; SAVE A REGISTER
8F AA 05E8 1013 5$: BICW #TTY$M_TANK_STOP-UCB$W_TT_HOLD(R5) ; RESET STOP CONDITIONS
15 0108 C5 0B E0 05EF 1014 BBS #TTY$V_TANK_BURST,UCB$W_TT_HOLD(R5),20$ ; BURST IN PROGRESS
05F5 1015 10$: TIMSET 1 ; CHAR IN TANK OR OTHER
05F5 1016 BRB 30$
1F 11 060A 1017 20$: MOVZWL UCB$W_TT_OUTLEN(R5),R1 ; GET NUMBER CHARACTERS
51 0120 C5 3C 060A 1020 TIMSET R1,R1
060F 1021 30$:
0629 1022 BBS #UCB$V_INT,UCB$W_STS(R5),40$ ; SKIP IF OUTPUT ON
13 64 A5 01 E0 0629 1023 MOVL UCB$L_CRB(R5),R1 ; ACCESS CRB ADDRESS
51 24 A5 D0 062E 1024 MOVL @CRB$C_INTD+VEC$C_IDB(R1),R1 ; GET CSR ADDRESS
51 2C B1 D0 0632 1025 BISW UCB$W_TT_UNITBIT(R5),4(R1) ; ENABLE LINE
04 A1 0106 C5 A8 0636 1026 BBSS #UCB$V_INT,UCB$W_STS(R5),40$ ; SHOW OUTPUT ACTIVE
00 64 A5 01 E2 063C 1027
0641 1028 40$:
51 8ED0 0641 1029 POPL R1
05 0644 1030 RSB
0645 1031 .IF NDF DZV
0645 1032 .ENABLE LSB
0645 1033 ;
0645 1034 ; SCHEDULE XOFF TO BE SENT
0645 1035 ;
0645 1036 DZ32$XOFF:
0645 1037 ;
0645 1038 ; SCHEDULE XON TO BE SENT
0645 1039 ;
0645 1040 DZ32$XON:
0108 C5 0100 8F A8 0645 1041 BISW #TTY$M_TANK_PREMPT,UCB$W_TT_HOLD(R5) ; SCHEDULE XON
010A C5 53 90 064C 1042 MOVW R3,UCB$B_TT_PREMPT(R5) ; SAVE THE CHARACTER
1F 64 A5 01 E0 0651 1043 BBS #UCB$V_INT,UCB$W_STS(R5),10$ ; IN THE PREMPT SLOT
0656 1044 ; IF OUTPUT ACTIVE,
0656 1045 ; FINISHED
51 24 A5 BB 0656 1046 PUSHR #^M<R1,R4> ; SAVE REGISTERS
54 2C A1 D0 0658 1047 MOVL UCB$L_CRB(R5),R1 ; ACCESS CRB ADDRESS
51 64 D0 065C 1048 MOVL CRB$C_INTD+VEC$C_IDB(R1),R4 ; GET IDB ADDRESS
0E A4 0106 C5 88 0660 1049 MOVL (R4),R1 ; GET CSR ADDRESS
07 A1 0E A4 90 0663 1050 BISB UCB$W_TT_UNITBIT(R5),IDB$B_TT_ENABLE(R4) ; ENABLE LINE
00 64 A5 01 E2 0669 1051 MOVW IDB$B_TT_ENABLE(R4),7(R1)
066E 1052 POPR #^M<R1,R4>
0670 1053 BBSS #UCB$V_INT,UCB$W_STS(R5),10$ ; SHOW OUTPUT ACTIVE
05 0675 1054 10$:
0675 1055 RSB
0676 1056 .DISABLE LSB
```

```
0676 1057
0676 1058 : RESUME STOPPED OUTPUT
0676 1059 :
0676 1060 :
0676 1061 DZ32$RESUME:
010B C5 0200 12 BB 0676 1062 PUSHR #*M<R1,R4> ; SAVE REGISTERS
      8F AA 0678 1063 5$: BICW #TTY$M_TANK_STOP-
      15 010B C5 0B E0 067F 1064 UCB$W_TT_HOLD(R5) ; RESET STOP CONDITIONS
      1F 11 0685 1065 10$: BBS #TTY$V_TANK_BURST,UCB$W_TT_HOLD(R5),20$ ; BURST IN PROGRESS
      51 0120 C5 3C 0685 1066 TIMSET 1 ; CHAR IN TANK OR OTHER
      0698 1067 BRB 30$ ; TIME OUT
      069A 1069 20$: MOVZWL UCB$W_TT_OUTLEN(R5),R1
      069F 1071 TIMSET R1,R1 ; SET THE TIMER
      06B9 1072 30$: BBS #UCB$V_INT,UCB$W_STS(R5),40$ ; SKIP IF OUTPUT ON
      1B 64 A5 01 E0 06B9 1073 MOVL UCB$W_CRB(R5),R1 ; ACCESS CRB ADDRESS
      51 24 A5 D0 06BE 1074 MOVL CRB$W_INTD+VEC$W_IDB(R1),R4 ; GET IDB ADDRESS
      54 2C A1 D0 06C2 1075 MOVL (R4),R1 ; GET CSR ADDRESS
      0E A4 0106 C5 88 06C6 1076 BISB UCB$W_TT_UNITBIT(R5),IDB$B_TT_ENABLE(R4) ; ENABLE LINE
      07 A1 0E A4 90 06CF 1077 MOVB IDB$B_TT_ENABLE(R4),7(R1)
      00 64 A5 01 E2 06D4 1079 BBSS #UCB$V_INT,UCB$W_STS(R5),40$ ; SHOW OUTPUT ACTIVE
      12 BA 06D9 1080 40$: POPR #*M<R1,R4>
      05 05 06D9 1081 RSB
      06DB 1082
      06DC 1083 .ENDC
```

```
06DC 1085 .SBTTL OUTPUT INTERRUPT SERVICE
06DC 1086 ++
06DC 1087 DZ$INTOUT - DZ-11 OUTPUT INTERRUPT SERVICE
06DC 1088
06DC 1089 FUNCTIONAL DESCRIPTION:
06DC 1090
06DC 1091 THIS ROUTINE IS ENTERED WHEN THE DZ-11 FINDS A LINE ENABLED
06DC 1092 AND AN EMPTY UART. THE CORRESPONDING UCB IS FOUND AND
06DC 1093 ANY OUTSTANDING PORT OUTPUT IS DONE. WHEN ALL OUTSTANDING PORT
06DC 1094 OUTPUT IS COMPLETED, THE CLASS DRIVER IS CALLED TO RETURN THE NEXT
06DC 1095 CHARACTER OR STRING TO BE OUTPUT. IF NO MORE OUTPUT IS FOUND, THEN
06DC 1096 THE LINE IS DISBALED.
06DC 1097
06DC 1098 INPUTS:
06DC 1099
06DC 1100 SP(00) = ADDRESS OF THE IDB
06DC 1101
06DC 1102 IMPLICIT INPUTS:
06DC 1103
06DC 1104 R0,R1,R2,R3,R4,R5 SAVED ON THE STACK.
06DC 1105
06DC 1106 OUTPUTS:
06DC 1107
06DC 1108 THE INTERRUPT IS DISMISSED.
06DC 1109
06DC 1110 --
06DC 1111 DZ11_OUT_EXIT: ; EXIT OUTPUT INTERRUPT
06DC 1112 ADDL #4,SP ; REMOVE IDB ADDRESS
06DC 1113 MOVQ (SP)+,R0 ; RESTORE REGISTERS
06DC 1114 MOVQ (SP)+,R2
06DC 1115 MOVQ (SP)+,R4
06DC 1116 REI ; DISMISS INTERRUPT
06DC 1117
06DC 1118 DZ11$INTOUT:: ; DZ-11 OUTPUT INTERRUPT SERVICE
06DC 1119
06DC 1120 DZ11_OUT_LOOP:
06DC 1121 MOVL @ (SP),R4 ; GET THE IDB ADDRESS
06DC 1122 MOVL (R4),R0 ; GET THE CSR ADDRESS
06DC 1123
06DC 1124 ; GET THE LINE INFO FROM THE CSR
06DC 1125
06DC 1126
06DC 1127 MOVW (R0),R2 ; GET THE CSR VALUE
06DC 1128 BGEQ DZ11_OUT_EXIT ; NO MORE LINES
06DC 1129 ASHL #-8,R2,R2 ; GET THE LINE NUMBER
06DC 1130 BICL #^C<7>,R2
06DC 1131 MOVL IDB$L UCBLST(R4)[R2],R5 ; GET THE UCB ADDRESS
06DC 1132 BEQL DZ11_OUT_LOOP ; IF EQL THEN DISMISS
06DC 1133
06DC 1134 ;
06DC 1135
06DC 1136 CHECK FOR BURST ACTIVE ON LINE
06DC 1137
06DC 1138 CMPB #TTY$M_TANK_BURST@-8,- ; ONLY BURST ACTIVE?
06DC 1139 UCBSW TT_HOED+1(R5)
06DC 1140 BEQL DZ11_BURST ; YES, CONTINUE BURST
06DC 1141
06DC 1142 ;
06DC 1143
06DC 1144 LOOK FOR NEXT OUTPUT STATE IN TANK
```

5E 04 C0 06DC 1112 ADDL #4,SP ; EXIT OUTPUT INTERRUPT
50 8E 7D 06DC 1113 MOVQ (SP)+,R0 ; REMOVE IDB ADDRESS
52 8E 7D 06DC 1114 MOVQ (SP)+,R2 ; RESTORE REGISTERS
54 8E 7D 06DC 1115 MOVQ (SP)+,R4
02 06DC 1116 REI ; DISMISS INTERRUPT
06DC 1117
06DC 1118 DZ11\$INTOUT:: ; DZ-11 OUTPUT INTERRUPT SERVICE
06DC 1119
54 00 BE D0 06DC 1120 DZ11_OUT_LOOP:
50 64 D0 06DC 1121 MOVL @ (SP),R4 ; GET THE IDB ADDRESS
06DC 1122 MOVL (R4),R0 ; GET THE CSR ADDRESS
06DC 1123
06DC 1124 ; GET THE LINE INFO FROM THE CSR
06DC 1125
06DC 1126
52 60 B0 06DC 1127 MOVW (R0),R2 ; GET THE CSR VALUE
E7 18 06DC 1128 BGEQ DZ11_OUT_EXIT ; NO MORE LINES
52 52 FB 8F 78 06DC 1129 ASHL #-8,R2,R2 ; GET THE LINE NUMBER
52 FFFFFFFF 8F CA 06DC 1130 BICL #^C<7>,R2
55 18 A442 D0 06DC 1131 MOVL IDB\$L UCBLST(R4)[R2],R5 ; GET THE UCB ADDRESS
E1 13 06DC 1132 BEQL DZ11_OUT_LOOP ; IF EQL THEN DISMISS
0708 1133
0708 1134
0708 1135
0708 1136
0109 08 91 0708 1137 CMPB #TTY\$M_TANK_BURST@-8,- ; ONLY BURST ACTIVE?
C5 070A 1138 UCBSW TT_HOED+1(R5)
40 13 070D 1139 BEQL DZ11_BURST ; YES, CONTINUE BURST
070F 1140
070F 1141
070F 1142

```
53 0109 C5 06 00 EA 070F 1142
                        070F 1143 FFS #0,#6,UCBSW_TT_HOLD+1(R5),R3
                        0716 1144 CASE R3,TYPE=B,<=
                        0716 1145 DZ11_PREMPT,- ; DISPATCH
                        0716 1146 DZ11_STOP,- ; SEND PREMPT CHARACTER
                        0716 1147 DZ11_CHAR,- ; STOP OUTPUT
                        0716 1148 DZ11_BURST,- ; CHAR IN TANK
                        0716 1149 > ; BURST IN PROGRESS
                        0722 1150
                        0722 1151 : NO PENDING DATA - LOOK FOR NEXT CHARACTER
                        0722 1152 :
                        64 A5 03 8A 0722 1153 BICB #UCBSM_TIM!UCBSM_INT,UCBSW_STS(R5); CLEAR TIMEOUT AND EXPECTED
                        0726 1154 :
                        0726 1155 : CALL CLASS DRIVER FOR MORE OUTPUT
                        0726 1156 :
01 FF 8F 010C D5 16 0726 1157 JSB @UCBSL_TT_GETNXT(R5) ; GET THE NEXT CHARACTER
010B C5 8F 072A 1158 CASEB UCBSB_TT_OUTTYPE(R5),#-1,#1; OPTIMIZE FOR THE SINGLE
                        0731 1159 : CHARACTER CASE BY SETTING THE
                        0731 1160 : LIMIT TO 1
0017' 0731 1161 1$: .WORD DZ11_START_BURST-1$ ; BURST SPECIFIED
000A' 0733 1162 .WORD 50$-1$ ; NONE
                        0735 1163 :
                        0735 1164 : OUTPUT A CHARACTER TO THE DZ-11
                        0735 1165 :
06 A0 53 9B 0735 1166 20$: MOVZBW R3,6(R0) ; OUTPUT CHARACTER
AE 11 0739 1167 BRB DZ11_OUT_LOOP
                        073B 1168 :
                        073B 1169 : DISABLE OUTPUT ON THIS LINE
                        073B 1170 :
                        073B 1171 50$:
A9 64 A5 01 E0 073B 1172 BBS #UCBSV_INT,- ; IF INT EXP, THEN DON'T RESET,
073D 1173 UCBSW_STS(R5),DZ11_OUT_LOOP ; COULD HAVE BEEN SET DURING CALLBACK
0740 1174 :
0740 1175 :
04 A0 0106 C5 AA 0740 1176 BICW UCBSW_TT_UNITBIT(R5),4(R0) ; RESET THE OUTPUT ENABLE
A1 11 0746 1177 BRB DZ11_OUT_LOOP
                        0748 1178 :
                        0748 1179 :
0800 8F AB 0748 1180 DZ11_START_BURST:
0108 C5 0748 1181 BISW #TTY$M_TANK_BURST,- ; SIGNAL BURST ACTIVE
                        074C 1182 UCBSW_TT_HOLD(R5)
                        074F 1183 :
                        074F 1184 : CONTINUE BURST OUTPUT
                        074F 1185 :
011C D5 90 074F 1186 DZ11_BURST:
06 A0 074F 1187 MOVB @UCBSL_TT_OUTADR(R5),- ; OUTPUT NEXT BYTE
011C C5 D6 0753 1188 6(R0)
0120 C5 B7 0755 1189 INCL UCBSL_TT_OUTADR(R5) ; UPDATE POINTER
                        0759 1190 DECW UCBSW_TT_OUTLEN(R5) ; UPDATE COUNT
                        075D 1191 BNEQ DZ11_OUT_LOOP ; NOT LAST CHARACTER
0800 8F AA 075F 1192 BICW #TTY$M_TANK_BURST,- ; RESET BURST ACTIVE
0108 C5 0763 1193 UCBSW_TT_HOLD(R5)
FF80 31 0766 1194 BRW DZ11_OUT_LOOP
                        0769 1195 :
                        0769 1196 : OUTPUT SINGLE CHARACTER
                        0769 1197 :
                        0769 1198 DZ11_CHAR:
```

```
06 A0 0108 C5 90 0769 1199      MOVB   UCBSW_TT_HOLD(R5),6(R0) ; OUTPUT CHAR IN TANK
      0400 8F AA 076F 1200      BICW   #TTYSM_TANK_HOLD,-      ; SHOW TANK EMPTY
      0108 C5      0773 1201      UCBSW_TT_HOLD(R5)
      FF70 31 0776 1202      BRW     DZ11_OUT_LOOP
      0779 1203      ;
      0779 1204      ; STOP THE OUTPUT
      0779 1205      ;
      0779 1206      DZ11_STOP:
      03      8A 0779 1207      BICB   #UCBSM_INT!UCBSM_TIM,-
      64 A5 077B 1208      UCBSW_STS(R5)
04 A0 0106 C5 AA 077D 1209      BICW   UCBSW_TT_UNITBIT(R5),4(R0) ; RESET OUTPUT ACTIVE
      FF63 31 0783 1210      BRW     DZ11_OUT_LOOP ; RESET THE OUTPUT ENABLE
      0786 1211      .ENABLE LSB
      0786 1212      ;
      0786 1213      ; SEND XON OR XOFF CHARACTER
      0786 1214      ;
      0786 1215      ;
      0786 1216      DZ11_PREMPT:
      0786 1217      BICW   #TTYSM_TANK_PREMPT,-      . RESET XOFF STATE
      0100 8F AA 0786 1218      UCBSW_TT_HOLD(R5)
      0108 C5      078A 1219      UCBSW_TT_PREMPT(R5),6(R0); OUTPUT CHARACTER
06 A0 010A C5 90 078D 1220      MOVB
      FF53 31 0793 1221      BRW     DZ11_OUT_LOOP
      0796 1222      .DISABLE      [SB
      0796 1223
```

```
0796 1225 .IF NDF DZV
0796 1226
0796 1227 : DZ-32 OUTPUT INTERRUPT SERVICE CODE
0796 1228
0796 1229
0796 1230 DZ32_OUT_EXIT:
0796 1231 ADDL #4,SP ; EXIT OUTPUT INTERRUPT
50 04 C0 0796 1231 MOVQ (SP)+,R0 ; REMOVE IDB ADDRESS
50 8E 7D 0799 1232 MOVQ (SP)+,R2 ; RESTORE REGISTERS
52 8E 7D 079C 1233 MOVQ (SP)+,R4
54 8E 7D 079F 1234 REI
02 07A2 1235
07A3 1236
07A3 1237 DZ32$INTOUT::
07A3 1238 DZ32_OUT_LOOP:
54 00 BE D0 07A3 1239 MOVL @ (SP),R4 ; GET THE IDB ADDRESS
50 64 D0 07A7 1240 MOVL (R4),R0 ; GET THE CSR ADDRESS
07AA 1241
07AA 1242 : GET THE LINE INFO FROM THE CSR
07AA 1243
07AA 1244
52 60 B0 07AA 1245 MOVW (R0),R2 ; GET THE CSR VALUE
52 E7 18 07AD 1246 BGEQ DZ32_OUT_EXIT
52 52 FB 8F 78 07AF 1247 ASHL #-8,R2,R2 ; GET THE LINE NUMBER
52 FFFFFFFF 8F CA 07B4 1248 BICL #~C<7>,R2
55 18 A442 D0 07B8 1249 MOVL IDB$UCBLST(R4)[R2],R5 ; GET THE UCB ADDRESS
E1 13 07C0 1250 BEQL DZ32_OUT_LOOP ; IF EQL THEN DISMISS
07C2 1251
07C2 1252 : CHECK FOR BURST ACTIVE ON LINE
07C2 1253
07C2 1254 CMPB #TTY$M_TANK_BURST-8,- ; ONLY BURST ACTIVE?
0109 C5 91 07C4 1255 UCBSW TT_HOLD+1(R5)
07C7 1256 BEQL DZ32_BURST ; YES, CONTINUE BURST
07C9 1257
07C9 1258 : LOOK FOR NEXT OUTPUT STATE IN TANK
07C9 1259
07C9 1260
53 0109 C5 06 00 EA 07C9 1261 FFS #0,#6,UCBSW TT_HOLD+1(R5),R3
07D0 1262 CASE R3,TYPE=B,<= ; DISPATCH
07D0 1263 DZ32_PREMPT,- ; SEND PREMPT CHARACTERS
07D0 1264 DZ32_STOP,- ; STOP OUTPUT
07D0 1265 DZ32_CHAR,- ; CHAR IN TANK
07D0 1266 DZ32_BURST,- ; BURST IN PROGRESS
07D0 1267 >
07DC 1268
07DC 1269 : NO PENDING DATA - LOOK FOR NEXT CHARACTER
07DC 1270
64 A5 03 8A 07DC 1271 BICB #UCBSM_TIM!UCBSM_INT,UCBSW_STS(R5); CLEAR TIMEOUT AND EXPECTED
07E0 1272
07E0 1273 : CALL CLASS DRIVER FOR MORE OUTPUT
07E0 1274
07E0 1275 JSB @UCBSL TT_GETNXT(R5) ; GET THE NEXT CHARACTER
01 FF 8F 010C D5 16 07E4 1276 CASEB UCBSB TT_OUTYPE(R5),#-1,#1 ; OPTIMIZE FOR THE SINGLE
07EB 1277 : CHARACTER CASE BY SETTING THE
07EB 1278 : LIMIT TO 1
0022 07EB 1279 1$: .WORD DZ32_START_BURST-1$ ; BURST SPECIFIED
000B 07ED 1280 .WORD 508-T$ ; NONE
07EF 1281 :
```

```
06 A0 53 90 07EF 1282 : OUTPUT A CHARACTER TO THE DZ-32
      FFAD 31 07EF 1283 :
      07EF 1284 20$: MOV B R3,6(R0) : OUTPUT CHARACTER
      07F3 1285 BRW DZ32_OUT_LOOP
      07F6 1286 :
      07F6 1287 : DISABLE OUTPUT ON THIS LINE
      07F6 1288 :
      07F6 1289 :
      07F6 1290 50$:
      07F6 1291 BBS #UCBSM_INT,- : IF INT EXP, THEN DON'T RESET,
      07F8 1292 UCBSM_STS(R5),DZ32_OUT_LOOP : COULD HAVE BEEN SET DURING CALLBACK
      07FB 1293 :
      07FB 1294 :
      54 00 BE D0 07FB 1295 MOVL @ (SP),R4 : GET IDB ADDRESS
      0106 C5 8A 07FF 1296 BICB UCBSM_TT_UNITBIT(R5),- : RESET THE OUTPUT ENABLE
      0E A4 0803 1297 IDBSM_TT_ENABLE(R4)
      07 A0 0E A4 90 0805 1298 MOV B IDBSM_TT_ENABLE(R4),7(R0)
      FF96 31 080A 1299 BRW DZ32_OUT_LOOP
      080D 1300 :
      080D 1301 :
      080D 1302 DZ32_START BURST:
      080D 1303 BISW #TTYSM_TANK_BURST,- : SIGNAL BURST ACTIVE
      0108 C5 AB 0811 1304 UCBSM_TT_HOLD(R5)
      0814 1305 DZ32_BURST:
      011C D5 90 0814 1306 MOVL @UCBSM_TT_OUTADR(R5),- : OUTPUT NEXT BYTE
      06 A0 0818 1307 6(R0)
      011C C5 D6 081A 1308 INCL UCBSM_TT_OUTADR(R5) : UPDATE POINTER
      0120 C5 B7 081E 1309 UCBSM_TT_OUTLEN(R5) : UPDATE COUNT
      07 12 0822 1310 BNEQ 60$ : NOT LAST CHARACTER
      0800 8F AA 0824 1311 BICW #TTYSM_TANK_BURST,- : RESET BURST ACTIVE
      0108 C5 0828 1312 UCBSM_TT_HOLD(R5)
      FF75 31 082B 1313 60$: BRW DZ32_OUT_LOOP
      082E 1314 :
      082E 1315 : OUTPUT SINGLE CHARACTER
      082E 1316 :
      082E 1317 DZ32_CHAR:
      06 A0 0108 C5 90 082E 1318 MOVL UCBSM_TT_HOLD(R5),6(R0) : OUTPUT CHAR IN TANK
      0400 8F AA 0834 1319 BICW #TTYSM_TANK_HOLD,- : SHOW TANK EMPTY
      0108 C5 0838 1320 UCBSM_TT_HOLD(R5)
      FF65 31 083B 1321 BRW DZ32_OUT_LOOP
      083E 1322 :
      083E 1323 : STOP OUTPUT
      083E 1324 :
      083E 1325 :
      083E 1326 DZ32_STOP:
      0840 1327 BICB #UCBSM_INT!UCBSM_TIM,- : RESET OUTPUT ACTIVE
      0842 1328 UCBSM_STS(R5) : GET IDB ADDRESS
      54 00 BE D0 0842 1328 MOVL @ (SP),R4 : GET IDB ADDRESS
      0106 C5 8A 0846 1329 BICB UCBSM_TT_UNITBIT(R5),- : RESET THE OUTPUT ENABLE
      0E A4 084A 1330 IDBSM_TT_ENABLE(R4)
      07 A0 0E A4 90 084C 1331 MOV B IDBSM_TT_ENABLE(R4),7(R0)
      FF4F 31 0851 1332 BRW DZ32_OUT_LOOP
      0854 1333 :
      0854 1334 : .ENABLE LSB
      0854 1335 :
      0854 1336 : SEND XON OR XOFF
      0854 1337 :
      0854 1338 DZ32_PREMPT:
```

```
06 A0 0100 8F AA 0854 1339 BICW #TTY$M TANK PREMT,- ; RESET XOFF STATE
      0108 C5 0858 1340 UCBSW TT_HOLD(R5)
      010A C5 90 085B 1341 MOVB UCBSB TT_PREMT(R5),6(R0); OUTPUT CHARACTER
      FF3F 31 0861 1342 BRW DZ32_OUT_LOOP
      0864 1343
      0864 1344
      0864 1345
      0864 1346 .ENDC .DISABLE LSB
```

```
0864 1348 .SBTTL SET SPEED, PARITY PARAMETERS
0864 1349
0864 1350 :++
0864 1351 :DZ$SET_LINE - RESET SPEED, PARITY
0864 1352 :
0864 1353 :FUNCTIONAL DESCRIPTION:
0864 1354 :
0864 1355 :INPUTS:
0864 1356 :
0864 1357 :R5 - UCB ADDRESS
0864 1358 :
0864 1359 :OUTPUTS:
0864 1360 :
0864 1361 :R4 USED
0864 1362 :--
0864 1363
0864 1364 DZ$SET_LINE:
0864 1365 MOVL UCB$$_CRB(R5),R4 ; ADDRESS CRB
0868 1366 :
0868 1367 :SET UP LINE SPEED AND PARITY
0868 1368 :
0868 1369 MOVL @CRB$$_INTD+VEC$_IDB(R4),R4 ; GET THE CSR ADDRESS VIA CRB
086C 1370 CLRL -(SP) ; RESET A TEMPORARY LOCATION
086E 1371 SUBB3 #1,UCB$_TT_SPEED(R5),1(SP) ; ADJUST DATA BASE SPEED
0875 1372 MOVB UCB$_TT_PARITY(R5),(SP) ; SET PARITY, STOP, CHARACTER SIZE
087A 1373 BICW #^XF007,(SP) ; CLEAR SPECIAL FIELDS
087F 1374 BBS #UCB$_TT_DSBL,- ; SKIP CLOCK ENABLE IF LINE DISABLED
0881 1375 UCB$_TT_MAINT(R5),3$
0885 1376 BISW #<DZLPR$_CLOCK>,(SP)
088A 1377 3$:
088A 1378 BISW UCB$_UNIT(R5),(SP) ; SET LINE NUMBER
088E 1379 BITW #DZCSR$_MODE,(R4) ; DZ32 CONTROLLER?
0891 1380 BNEQ 10$ ; YES
0893 1381 5$:
0893 1382 CVTLW (SP)+,2(R4) ; INSERT AS LINE PARAMETER
0897 1383 RSB
0898 1384
0898 1385 :HANDLE DZ-32 SPECIFIC FUNCTIONS
0898 1386 10$:
0898 1387 CMPB UCB$_TT_SPEED(R5),- ; TRANSMIT/RECEIVE THE SAME
089C 1388 UCB$_TT_SPEED+1(R5)
089F 1389 BEQL 5$ ; YES, NO SPLIT SPEED
08A1 1390 TSTB UCB$_TT_SPEED+1(R5) ; RECEIVE SPEED SPECIFIED?
08A5 1391 BEQL 5$ ; NO, NO SPLIT SPEED
08A7 1392
08A7 1393 :SET SPLIT SPEED
08A7 1394
08A7 1395 BISW #DZLPR$_SPLIT,(SP) ; SET SPLIT SPEED BIT
08AC 1396 BRB 5$ ; COMPLETE SETUP
08AE 1397
```

01 AE 54 24 A5 D0 00F4 C5 01 83 086C 1370
6E 00F8 C5 90 0875 1372
6E F007 8F AA 087A 1373
05 012A C5 E0 087F 1374
6E 1000 8F AB 0885 1376
6E 54 A5 AB 088A 1378
64 01 B3 088E 1379
05 12 0891 1380
02 A4 8E F7 0893 1381
05 0893 1382
0897 1383
0898 1384
0898 1385
0898 1386
00F4 C5 91 0898 1387
00F5 C5 089C 1388
F2 13 089F 1389
00F5 C5 95 08A1 1390
EC 13 08A5 1391
08A7 1392
08A7 1393
08A7 1394
6E 2000 8F AB 08A7 1395
E5 11 08AC 1396
08AE 1397

```
08AE 1399
08AE 1400 .SBTTL INITIALIZE DZ-11 MODEM POLLING
08AE 1401
08AE 1402 :++
08AE 1403 : DZ$SET_MODEM - INIT MODEM POLLING
08AE 1404 :
08AE 1405 : FUNCTIONAL DESCRIPTION:
08AE 1406 :
08AE 1407 : INIT DZ-11 MODEM TRANSITION POLLING IF NOT ALREADY ACTIVE. LINK CRB
08AE 1408 : FOR CURRENT LINE INTO MODEM TRANSITION POLLING LIST
08AE 1409 :
08AE 1410 : INPUTS:
08AE 1411 :
08AE 1412 : R5 - UCB ADDRESS
08AE 1413 :
08AE 1414 : OUTPUTS:
08AE 1415 :
08AE 1416 : R0-R4 USED
08AE 1417 :--
08AE 1418
08AE 1419 DZ11$SET MODEM:
54 24 A5 D0 08AE 1420 MOVL UCB$$_CRB(R5),R4 : ADDRESS CRB
00000000'EF D5 08B2 1421 TSTL DZ$_DIALUP : DZ-11 POLLING ALREADY ACTIVE?
18 12 08B8 1422 BNEQ 5$ : YES, SKIP STARTUP
38 BB 08BA 1423 PUSHR #M<R3,R4,R5>
55 00000004'EF DE 08BC 1424 MOVAL DZ$TIMQUENT,R5 : ADDRESS OF TIMER ENTRY
0B A5 06 90 08C3 1425 MOVB #IPL$_QUEUEAST,TQESB_RQTYPE(R5) : SET FORK IPL
000008D0'EF 9F 08C7 1426 PUSHAB 4$ : RETURN ADDRESS
0021 31 08CD 1427 BRW 30$ : QUEUE FORK
38 BA 08D0 1428 4$: POPR #M<R3,R4,R5>
5$ 08D2 1429 5$:
53 18 A4 DE 08D2 1430 MOVAL CRB$_DZ_MODEM(R4),R3 : ADDRESS OF DZ CRB THREAD
51 00000000'EF DE 08D6 1431 MOVAL DZ$_DIALUP,R1 : ADDRESS OF DZ TIMER LIST HEAD
52 51 D0 08D0 1432 MOVL R1,R2
08E0 1433 :
08E0 1434 : LINK CRB INTO DZ-11 MODEM POLLER LIST IF NEEDED
08E0 1435 :
08E0 1436 10$:
53 62 D1 08E0 1437 CMPL (R2),R3 : IS CRB ON LIST
0B 13 08E3 1438 BEQL 20$ : YES, DONE
52 62 D0 08E5 1439 MOVL (R2),R2 : POINT TO NEXT CRB
F6 12 08E8 1440 BNEQ 10$ : LOOK FOR NEXT
63 61 D0 08EA 1441 MOVL (R1),(R3) : LINK CRB AT LIST HEAD
61 53 D0 08ED 1442 MOVL R3,(R1) :
05 08F0 1443 20$: RSB
08F1 1445 30$:
00000000'GF 16 08F1 1446 JSB G$EXES$FORK : FORK TO QUEUE TIMER ENTRY
08F7 1447 DSBINT #IPL$_SYNCH
0C A5 FA45 CF 9E 08FD 1448 MOVAB W'DZ$TIMER,TQES$_FPC(R5) : ADDRESS OF TIMER SERVICE ROUTINE
20 A5 00000000'GF D0 0903 1449 MOVL G$TTYSGL_$DELTA,TQESQ_$DELTA(R5) : INTERVAL IS SYSGEN PARAMETER
090B 1450 :
0B A5 05 90 090B 1451 MOVB #TQES$_SSREPT,TQESB_RQTYPE(R5)
50 00000000'GF 7D 090F 1452 MOVQ G$EXES$Q_SYSTIME,R0
50 00000000'GF C0 0916 1453 ADDL G$TTYSGL_$DELTA,R0
51 00 08 091D 1454 ADWC #0,R1
00000000'GF 16 0920 1455 JSB G$EXES$INSTIME : INSERT INTO TIMER QUEUE
```

DZDRIVER
V04-000

- Port Driver for DZ-11 support
INITIALIZE DZ-11 MODEM POLLING

L 14

16-SEP-1984 02:24:50 VAX/VMS Macro V04-00
5-SEP-1984 04:15:55 [TTDRVR.SRC]DZDRIVER.MAR;1

Page 34
(1)

05 0926 1456 ENBINT
0929 1457 RSB
092A 1458

: RESTORE IPL

```
092A 1460
00000000 1461 .PSECT $$$117_DATA,QUAD
0000 1462 :
0000 1463 : DZ-11 TIMER DATA STRUCTURES FOR DETECTION OF DZ-11 MODEM TRANSITIONS
0000 1464 :
0000 1465 :
0000 1466 DZSL_DIALUP:
00000000 0000 1467 .LONG 0 ; LINKED LIST OF DZ-11 CRB'S
0004 1468 ; USING MODEM CONTROL
0004 1469 DZSTIMQUENT: ; TIMER QUE ENTRY USED TO
0004 1470 ; SAMPLE DZ-11 MODEM SIGNALS
0004 1471 ; ON PERIODIC BASIS
0004 1472
00000038 0008 1473 .ALIGN QUAD
0038 1474 .BLKB TQESC_LENGTH
0038 1475 STO_TQE TQESW_SIZE,WORD,TQESC_LENGTH,DZSTIMQUENT
0038 1476 STO_TQE TQESB_TYPE,BYTE,DYN$C-TQE,DZSTIMQUENT
0038 1477 STO_TQE TQESB_RQTYPE,BYTE,TQESC_SSREPT,DZSTIMQUENT
0038 1478
0038 1479 DZSEND: ; End of driver
0038 1480
0038 1481 .END
```

DZDRIVER
Symbol table

- Port Driver for DZ-11 support

N 14

16-SEP-1984 02:24:50 VAX/VMS Macro V04-00
5-SEP-1984 04:15:55 [TTDRVR.SRC]DZDRIVER.MAR;1

Page 36
(1)

```

$$$ = 00000020 R 02
$$$$$ = 00000038 R 04
$$OP = 00000002
ATS_UBA = 00000001
BIT... = 00000010
CLASS_DDT = 00000010
CLASS_DS_TRAN = 0000000C
CLASS_GETNXT = 00000000
CLASS_POWERFAIL = 00000020
CLASS_PUTNXT = 00000004
CLASS_READERROR = 00000014
CLASS_SETUP_UCB = 00000008
CRBSB_DZ_CARRIER = 0000001D
CRBSB_DZ_DTR = 0000001E
CRBSB_DZ_RING = 0000001C
CRBSB_TT_TYPE = 0000000B
CRBSL_DZ_MODEM = 00000018
CRBSL_INTD = 00000024
CRBSL_INTD2 = 00000048
DCS_TERM = 00000042
DDBSL_DDT = 0000000C
DEVSM_AVL = 00040000
DEVSM_CCL = 00000002
DEVSM_IDV = 04000000
DEVSM_NNM = 00000200
DEVSM_ODV = 08000000
DEVSM_REC = 00000001
DEVSM_TRM = 00000004
DPTSC_LENGTH = 00000038
DPTSC_VERSION = 00000004
DPTSINITAB = 00000038 R 02
DPTSM_NOUNLOAD = 00000004 R 02
DPTSREINITAB = 000000D3 R 02
DPTSTAB = 00000000 R 02
DPTSM_VECTOR = 0000001E
DTS_DZ11 = 00000042
DTS_DZ32 = 00000043
DYNBS_CRB = 00000005
DYNBS_DDB = 00000006
DYNBS_DPT = 0000001E
DYNBS_ORB = 00000049
DYNBS_TQE = 0000000F
DYNBS_UCB = 00000010
DZ$ABORT = 000005CC R 03
DZ$CTRL_ERROR = 0000014A R 03
DZ$DDT = 00000000 R 03
DZ$DPT = 00000000 R 02
DZ$END = 00000038 R 04
DZ$INITIAL = 000000AD R 03
DZ$INITLINE = 0000014B R 03
DZ$SL_DIALUP = 00000000 R 04
DZ$SNOLL = 000000AC R 03
DZ$SET_LINE = 00000864 R 03
DZ$STOP = 000005C4 R 03
DZ$TIMER = 00000346 R 03
DZ$TIMQUENT = 00000004 R 04
DZ$UNIT_ERROR = 00000257 R 03

```

```

DZ11$DS_SET = 000002B5 R 03
DZ11$INTINP = 000003E9 R 03
DZ11$INTOUT = 000006E9 R 03
DZ11$MAINT = 0000028C R 03
DZ11$RESUME = 000005E6 R 03
DZ11$SET_MODEM = 000008AE R 03
DZ11$STARTIO = 00000546 R 03
DZ11$VEC = 00000038 R 03
DZ11$VECEND = 00000070 R 03
DZ11$XOFF = 0000059A R 03
DZ11$XON = 0000059A R 03
DZ11_BURST = 0000074F R 03
DZ11_CHAR = 00000769 R 03
DZ11_OUT_EXIT = 000006DC R 03
DZ11_OUT_LOOP = 000006E9 R 03
DZ11_PREMPT = 00000786 R 03
DZ11_START_BURST = 00000748 R 03
DZ11_STOP = 00000779 R 03
DZ32$DS_SET = 000002ED R 03
DZ32$INTINP = 00000477 R 03
DZ32$INTOUT = 000007A3 R 03
DZ32$MAINT = 0000025C R 03
DZ32$RESUME = 00000676 R 03
DZ32$STARTIO = 0000056C R 03
DZ32$VEC = 00000070 R 03
DZ32$VECEND = 000000A8 R 03
DZ32$XOFF = 00000645 R 03
DZ32$XON = 00000645 R 03
DZ32_BURST = 00000814 R 03
DZ32_CHAR = 0000082E R 03
DZ32_OUT_EXIT = 00000796 R 03
DZ32_OUT_LOOP = 000007A3 R 03
DZ32_PREMPT = 00000854 R 03
DZ32_START_BURST = 0000080D R 03
DZ32_STOP = 0000083E R 03
DZCSRSM_CLEAR = 00000010
DZCSRSM_DS_CHG = 00000800
DZCSRSM_DS_ENAB = 00000002
DZCSRSM_LINE = 00000700
DZCSRSM_MAINT = 00000008
DZCSRSM_MASTENAB = 00000020
DZCSRSM_MODE = 00000001
DZCSRSM_RCVINT = 00000040
DZCSRSM_RCVRDY = 00000080
DZCSRSM_SNDINT = 00004000
DZCSRSM_SNDRDY = 00008000
DZCSRSS_CLEAR = 00000001
DZCSRSS_DS_CHG = 00000001
DZCSRSS_DS_ENAB = 00000001
DZCSRSS_LINE = 00000003
DZCSRSS_MAINT = 00000001
DZCSRSS_MASTENAB = 00000001
DZCSRSS_MODE = 00000001
DZCSRSS_RCVINT = 00000001
DZCSRSS_RCVRDY = 00000001
DZCSRSS_SNDINT = 00000001
DZCSRSS_SNDRDY = 00000001

```

DZV
V04

DZDRIVER
Symbol table

- Port Driver for DZ-11 support

B 15

16-SEP-1984 02:24:50 VAX/VMS Macro V04-00
5-SEP-1984 04:15:55 [TTDRVR.SRC]DZDRIVER.MAR;1

Page 37
(1)

DZCSR\$V_CLEAR	= 00000004	EXESGL_ABSTIM	*****	X	03
DZCSR\$V_DS_CHG	= 0000000B	EXESGL_TENUSEC	*****	X	03
DZCSR\$V_DS_ENAB	= 00000001	EXESGL_UBDELAY	*****	X	03
DZCSR\$V_LINE	= 00000008	EXESGL_SYSTIME	*****	X	03
DZCSR\$V_MAINT	= 00000003	EXESINSTIMQ	*****	X	03
DZCSR\$V_MASTENAB	= 00000005	FUNCTAB_LEN	= 00000000		
DZCSR\$V_MODE	= 00000000	IDB\$B TT_ENABLE	= 0000000E		
DZCSR\$V_RCVINT	= 00000006	IDB\$B_UCBLST	= 00000018		
DZCSR\$V_RCVRDY	= 00000007	IOSM_LINE_OFF	= 00000200		
DZCSR\$V_SNDINT	= 0000000E	IOSM_LINE_ON	= 00000800		
DZCSR\$V_SNDRDY	= 0000000F	IOSM_LOOP	= 00000080		
DZLCS1\$M_ACK	= 00008000	IOSM_LOOP_EXT	= 00001000		
DZLCS1\$S_ACK	= 00000001	IOSM_UNLOOP	= 00000100		
DZLCS1\$V_ACK	= 0000000F	IOCS\$MNTVER	*****	X	03
DZLPR\$M_CLOCK	= 00001000	IOCS\$RETURN	*****	X	03
DZLPR\$M_LINE	= 00000007	IPL\$_QUEUEAST	= 00000006		
DZLPR\$M_ODD	= 00000080	IPL\$-SYNCH	= 00000008		
DZLPR\$M_PARITY	= 00000040	MODEM\$C_DATASET	= 00000003		
DZLPR\$M_SIZE	= 00000018	MODEM\$C_INIT	= 00000000		
DZLPR\$M_SPEED	= 00000F00	ORBS\$B_FLAGS	= 0000000B		
DZLPR\$M_SPLIT	= 00002000	ORBS\$B_OWNER	= 00000000		
DZLPR\$M_STOP	= 00000020	ORBS\$M_PROT_16	= 00000001		
DZLPR\$S_CLOCK	= 00000001	ORBS\$W_PROT	= 00000018		
DZLPR\$S_LINE	= 00000003	PORT_ABORT	= 00000020		
DZLPR\$S_ODD	= 00000001	PORT_DS_SET	= 0000000C		
DZLPR\$S_PARITY	= 00000001	PORT_LENGTH	= 00000038		
DZLPR\$S_SIZE	= 00000002	PORT_MAINT	= 00000030		
DZLPR\$S_SPEED	= 00000004	PORT_RESUME	= 00000024		
DZLPR\$S_SPLIT	= 00000001	PORT_SET_LINE	= 00000008		
DZLPR\$S_STOP	= 00000001	PORT_SET_MODEM	= 00000028		
DZLPR\$V_CLOCK	= 0000000C	PORT_STARTIO	= 00000000		
DZLPR\$V_LINE	= 00000000	PORT_STOP	= 00000018		
DZLPR\$V_ODD	= 00000007	PORT_VECTOR	= 00000038	R	03
DZLPR\$V_PARITY	= 00000006	PORT_XOFF	= 00000014		
DZLPR\$V_SIZE	= 00000003	PORT_XON	= 00000010		
DZLPR\$V_SPEED	= 00000008	PR\$_IPL	= 00000012		
DZLPR\$V_SPLIT	= 00000000	SIZ_...	= 00000001		
DZLPR\$V_STOP	= 00000005	SS\$ NORMAL	= 00000001		
DZRCV\$M_BUF	= 000000FF	TQES\$B_RQTYPE	= 0000000B		
DZRCV\$M_FRAMER	= 00002000	TQES\$B_TYPE	= 0000000A		
DZRCV\$M_LINE	= 00000700	TQES\$C_LENGTH	= 00000030		
DZRCV\$M_OVERRUN	= 00004000	TQES\$C_SSREPT	= 00000005		
DZRCV\$M_PARERR	= 00001000	TQES\$L_FPC	= 0000000C		
DZRCV\$M_VALID	= 00008000	TQES\$Q_DELTA	= 00000020		
DZRCV\$S_BUF	= 00000008	TQES\$W_SIZE	= 00000008		
DZRCV\$S_FRAMER	= 00000001	TT\$M_DS_CTS	= 00000010		
DZRCV\$S_LINE	= 00000003	TT\$M_DS_DSR	= 00000080		
DZRCV\$S_OVERRUN	= 00000001	TT\$V_DS_CARRIER	= 00000005		
DZRCV\$S_PARERR	= 00000001	TT\$V_DS_DTR	= 00000001		
DZRCV\$S_VALID	= 00000001	TT\$V_DS_RING	= 00000006		
DZRCV\$V_BUF	= 00000000	TT\$V_MODEM	= 00000015		
DZRCV\$V_FRAMER	= 0000000D	TT\$ UNKNOWN	= 00000000		
DZRCV\$V_LINE	= 00000008	TTY\$GB_DEFSPEED	*****	X	02
DZRCV\$V_OVERRUN	= 0000000E	TTY\$GB_PARITY	*****	X	02
DZRCV\$V_PARERR	= 0000000C	TTY\$GB_RSPEED	*****	X	02
DZRCV\$V_VALID	= 0000000F	TTY\$GL_DEFCHAR	*****	X	02
EXES\$FORR	*****	TTY\$GL_DEFCHAR2	*****	X	02

X 03

DZV
V04

DZDRIVER
Symbol table

- Port Driver for DZ-11 support

C 15

16-SEP-1984 02:24:50 VAX/VMS Macro V04-00
5-SEP-1984 04:15:55 [TTDRVR.SRC]DZDRIVER.MAR;1

Page 38
(1)

TTYSGL_DELTA	*****	X	03	UCBSW_TT_UNITBIT	= 00000106
TTYSGL_DPT	*****	X	03	UCBSW_UNIT	= 00000054
TTYSGL_OWNUIC	*****	X	02	VECSL_IDB	= 00000008
TTYSGW_DEFBUF	*****	X	02	VECSL_INITIAL	= 0000000C
TTYSGW_PROT	*****	X	02	VECSL_UNITINIT	= 00000018
TTYSM_TANK_BURST	= 00000800				
TTYSM_TANK_HOLD	= 00000400				
TTYSM_TANK_PREMPT	= 00000100				
TTYSM_TANK_STOP	= 00000200				
TTYSV_PC_NOTIME	= 00000000				
TTYSV_TANK_BURST	= 0000000B				
UCBSB_DEVCLASS	= 00000040				
UCBSB_DEVTTYPE	= 00000041				
UCBSB_DIPL	= 0000005E				
UCBSB_FIPL	= 0000000B				
UCBSB_TT_DEPARI	= 000000EC				
UCBSB_TT_DETYPE	= 000000F0				
UCBSB_TT_DS_RCV	= 00000124				
UCBSB_TT_DS_TX	= 00000125				
UCBSB_TT_MAINT	= 0000012A				
UCBSB_TT_OUTYPE	= 0000010B				
UCBSB_TT_PARITY	= 000000F8				
UCBSB_TT_PREMPT	= 0000010A				
UCBSC_TT_LENGTH	= 00000134				
UCBSL_CRB	= 00000024				
UCBSL_DDB	= 00000028				
UCBSL_DDT	= 00000088				
UCBSL_DEVCHAR	= 00000038				
UCBSL_DEVCHAR2	= 0000003C				
UCBSL_DEVDEPEND	= 00000044				
UCBSL_DEVDEPND2	= 00000048				
UCBSL_DUETIME	= 0000006C				
UCBSL_TT_CLASS	= 00000114				
UCBSL_TT_DECHA1	= 000000C8				
UCBSL_TT_DECHAR	= 000000C4				
UCBSL_TT_GETNXT	= 0000010C				
UCBSL_TT_OUTADR	= 0000011C				
UCBSL_TT_PORT	= 00000118				
UCBSL_TT_PUTNXT	= 00000110				
UCBSL_TT_RTIMOU	= 000000B4				
UCBSL_TT_WBLINK	= 000000D0				
UCBSL_TT_WFLINK	= 000000CC				
UCBSM_INT	= 00000002				
UCBSM_ONLINE	= 00000010				
UCBSM_TIM	= 00000001				
UCBSM_TT_DSBL	= 00000080				
UCBSV_INT	= 00000001				
UCBSV_POWER	= 00000005				
UCBSV_TT_DSBL	= 00000007				
UCBSW_DEVBUFSIZ	= 00000042				
UCBSW_STS	= 00000064				
UCBSW_TT_DESIZE	= 000000F1				
UCBSW_TT_DESPEE	= 000000E8				
UCBSW_TT_HOLD	= 00000108				
UCBSW_TT_OUTLEN	= 00000120				
UCBSW_TT_PRTCTL	= 00000122				
UCBSW_TT_SPEED	= 000000F4				

DZV
V04

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$AB\$\$	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$105_PROLOGUE	0000000E (222.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
\$\$\$115_DRIVER	0000092A (2346.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG
\$\$\$117_DATA	00000038 (56.)	04 (4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC QUAD

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	30	00:00:00.07	00:00:00.37
Command processing	128	00:00:00.36	00:00:01.19
Pass 1	719	00:00:23.05	00:00:48.04
Symbol table sort	0	00:00:03.36	00:00:06.16
Pass 2	253	00:00:04.73	00:00:09.11
Symbol table output	35	00:00:00.20	00:00:00.26
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1169	00:00:31.80	00:01:05.15

The working set limit was 2250 pages.

187090 bytes (366 pages) of virtual memory were used to buffer the intermediate code.

There were 170 pages of symbol table space allocated to hold 3041 non-local and 103 local symbols.

1481 source lines were read in Pass 1, producing 25 object records in Pass 2.

72 pages of virtual memory were used to define 67 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
_\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	33
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	12
TOTALS (all libraries)	45

3540 GETS were required to define 45 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:DZDRIVER/OBJ=OBJ\$:DZDRIVER MSRC\$:DZDRIVER/UPDATE=(ENH\$:DZDRIVER)+EXECMLS/LIB

0402 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

